

STUDY OF TIME STEPPING METHODS FOR STIFF SYSTEMS OF DYNAMIC EQUATIONS

A DISSERTATION

Submitted in partial fulfilment of the requirements for the award of the degree

of

MASTER OF ENGINEERING

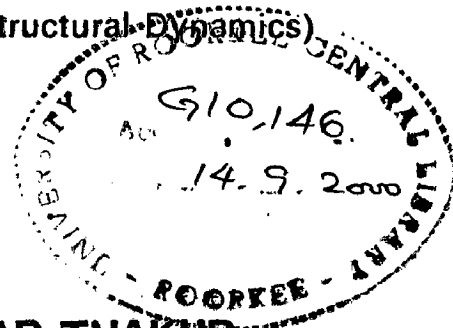
in

EARTHQUAKE ENGINEERING

(With Specialization in Structural Dynamics)

By

SANTOSH KUMAR THAKUR



DEPARTMENT OF EARTHQUAKE ENGINEERING
UNIVERSITY OF ROORKEE
ROORKEE-247 667 (INDIA)

JANUARY, 2000


CANDIDATE'S DECLARATION

I hereby declare that the work which is being presented in the dissertation titled "STUDY OF TIME STEPPING METHODS FOR STIFF SYSTEMS OF DYNAMIC EQUATIONS" in partial fulfillment of the requirements for the award of the degree of **MASTER OF ENGINEERING** in Earthquake Engineering, submitted to the Department of Earthquake Engineering, University of Roorkee, Roorkee, India, is the record of my own work carried out during the period from August, 1999 to January, 2000 under the supervision of Dr. G. I. Prajapati, Professor and Dr. Vipul Prakash, Asstt. Professor, Department of Earthquake Engineering, University of Roorkee, Roorkee.

The matter embodied in this dissertation has not been submitted for the award of any other degree.

Dated: 28/01/2000

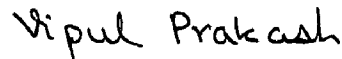
Place: Roorkee


Santosh kumar Thakur

This is to certify that the above statement made by the candidate is correct to the best of my knowledge.



Dr. G. I. Prajapati
Professor
Deptt. of Earthquake Engg.
University of Roorkee
Roorkee, India.



Dr. Vipul Prakash 28/1/2000
Asstt. Professor
Deptt. of Earthquake Engg.
University of Roorkee
Roorkee, India.

ACKNOWLEDGEMENT

I deeply acknowledged the understanding and guidance given to me by my guides, Dr. G. I. Prajapati, Professor and Dr. Vipul Prakash, Asstt. Professor, Department of Earthquake Engineering, University of Roorkee, Roorkee.

I am thankful to my guru Swami Sri Ramsurat Kumar whose divine presence guided me whenever I needed it.

I am extremely grateful to my parents and brothers for their support. I am indebted to my beloved junior Kumar Venkatesh for providing me his P.C when I was in extreme need of it. Last but not the least I am obliged to the faculty members and friends for helping me directly or indirectly in reaching this stage.

Santosh Thakur
(Santosh kumar Thakur)

ABSTRACT

An undamped stiff two degree of freedom system is considered for this study. The solution methods of equilibrium equation includes mode superposition and some popular time stepping methods. The time history for various time stepping methods are obtained by the aid of computer programming in C++. Due to inherent property of stiff system higher frequency is obtained of the order of 100 times greater than the fundamental frequency. This leads to the time history obtained in high frequency as well as low frequency region. An aim is to get the accurate response in low frequency mode. Another aim is to filter out response in high frequency mode in as few time steps as possible.

The popular time stepping methods are studied with respect to stability of solution, accuracy in first mode response and damping characteristics for higher mode response.

The solutions obtained by the popular time stepping methods for the two degree of freedom undamped system, are compared with the exact response obtained by mode superposition for the undamped case.

The strengths and weaknesses of the time stepping methods studied in chapter 4.

LIST OF SYMBOLS

A	:	amplification matrix
A_{ij}	:	i^{th} mode corresponding to j^{th} frequency
I	:	identity matrix
K	:	stiffness matrix of the system
L	:	load vector of the system
M	:	mass matrix of the system
M	:	m^{th} step in response
n	:	n^{th} step in response
T	:	time period of the system
Δt	:	time step
t_{cr}	:	critical time period of the system
U	:	displacement vector
ω	:	frequency of the system
λ	:	square of frequency of the system
ξ	:	damping of the system
$\rho(A)$:	spectral radius of matrix A

CONTENTS

Chapter	Title	Page No.
	CANDIDATE'S DECLARATION	
	ACKNOWLEDGEMENT	
	ABSTRACT	
	LIST OF SYMBOLS	
1.0	INTRODUCTION	1
2.0	MODE SUPERPOSITION METHOD	4
2.1	General	4
2.2	Analysis	
3.0	TIME STEPPING METHODS	6
3.1	General	6
3.2	Central difference method	6
3.3	Houbolt method	8
3.4	Newmark family of method	10
3.4.1	Newmark's constant average acceleration method	10
3.4.2	Newmark's linear acceleration method	11
3.5	Wilson theta method	12
4.0	STABILITY, ACCURACY AND DAMPING CHARACTERISTICS	15
4.1	Stability analysis (General)	15
4.2	Derivation of Amplification matrix	16
4.2.1	Central difference method	16
4.2.2	Houbolt method	17
4.2.3	Wilson theta method	17
4.2.4	Newmark method	19
4.3	Accuracy analysis	21

4.4	Numerical dissipation	22
4.5	Logarithmic damping	23
5.0	STUDY OF RESPONSE HISTORY	25
5.1	General	25
5.2	Central difference method	25
5.3	Trapezoidal method	26
5.4	Damped Newmark method	26
5.5	Wilson theta method	27
5.6	Houbolt method	27
	CONCLUSIONS	28
	SCOPE FOR FURTHER STUDIES	29
	REFERENCES	30
	FIGURES	31
	APPENDIX	45

INTRODUCTION

Two principal procedures are used for the solution of the dynamic equilibrium equation:

$$M\ddot{U} + KU = R$$

The two procedures are mode superposition and direct integration method. In the mode superposition method the response in the lowest few modes is usually evaluated. The response in the higher modes is considered to be an artifact of the mathematical modeling of the structure and is usually not evaluated and is ignored. Moreover the method can be used only for linear analysis.

Several time stepping schemes are popular for the direct integration method. These are the Central difference method, the Houbolt method, the Wilson-theta method and the Newmark family of methods including constant average acceleration (also called trapezoidal method) and linear acceleration method. These methods are reexamined in detail in this thesis. The desirable features of the time stepping methods are following:

- Unconditionally stable so that the time step size used is governed only by the time periods of the modes whose response is of interest and not by the time period of the highest mode present in the mathematical model.
- Accurate for obtaining the response in lower modes, i.e, for modes for which T (time period) is much greater than Δt (time step size). In other words $\Delta t \leq T_1 / n$ where T_1 is the highest ~~mode~~ ^{time period} of interest and $n \gg 1$ may be 4, 10 or 20 depending on accuracy.

- Must possess high numerical damping so that the responses in the modes higher than those of interest should be damped out.

~ determine t_{cr} ?

It is seen that using the central difference method, a time step Δt smaller than a critical time step t_{cr} has to be used; but employing the other three integration schemes, a similar time step limitation is not required. Hence all the considered methods except the Central difference method are unconditionally stable and meet the first desirable requirement.

An undamped stiff two-degree of freedom model having the following specifications is considered for studying the second and third desirable features.

$$\begin{bmatrix} m_1 & 0 \\ 0 & m_2 \end{bmatrix} \begin{bmatrix} \ddot{U}_1 \\ \ddot{U}_2 \end{bmatrix} + \begin{bmatrix} k_1 + k_2 & -k_2 \\ -k_2 & k_2 \end{bmatrix} \begin{bmatrix} U_1 \\ U_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

where, $m_1 = m_2 = 1$ and $k_1 = 10000, k_2 = 1$ with initial conditions as,

$$\begin{bmatrix} U_1 \\ U_2 \end{bmatrix} = \begin{bmatrix} 1 \\ 10 \end{bmatrix} \quad \text{and} \quad \begin{bmatrix} \dot{U}_1 \\ \dot{U}_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

The intent of two degree of freedom model considered is to represent the character of large system. The exact solution of the above equilibrium equation in chapter 2 shows that the first mode is intended to represent those modes of a large system that are physically important and must be accurately integrated. The second mode represents the high frequency mode of a large system in which significant response is not expected. It is desirable that the step by step integrator filters out these high modes from the system.

An important observation was that the cost of direct integration analysis is directly proportional to the number of time steps required for the solution. It follows that the selection of appropriate time step size is of much importance. On one hand time step must be small

enough to obtain accuracy in the solution; but on the other hand time step must not be smaller than necessary because with such time step the solution is more costly than what is required. For this reason desired time step size $\Delta t \approx T_1 / 20 \approx 5T_2$ to obtain accuracy in computation of response in first mode. The two fundamental concepts in selecting an appropriate time step Δt are stability and accuracy characteristics of integration methods, which ^{are} ~~is~~ discussed later on.

Finally comparison study is done for damping out of higher modes by all integration methods and damping phenomenon by considering logarithmic damping.

MODE SUPERPOSITION METHOD

2.1 GENERAL

The response analysis by mode superposition method requires the following [10]:

- The solution of eigen values and eigen vectors of the problem.
- The solution of the decoupled equilibrium equations and
- Finally the superposition of the response in each eigen vectors.

2.2 ANALYSIS

An undamped stiff two degree of freedom system is considered for the analysis.

$$M\ddot{U} + KU = 0 \quad (2.1)$$

The solution can be postulated to be of the form $U = A \sin(\omega t + \psi)$. The following case is considered for the analysis:

$$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \ddot{U}_1 \\ \ddot{U}_2 \end{bmatrix} + \begin{bmatrix} 10001 & -1 \\ -1 & 1 \end{bmatrix} \begin{bmatrix} U_1 \\ U_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \quad (2.2)$$

with the initial conditions as,

$$\begin{bmatrix} U_1 & U_2 \end{bmatrix}^T = \begin{bmatrix} 1 & 10 \end{bmatrix}^T \quad \begin{bmatrix} \dot{U}_1 & \dot{U}_2 \end{bmatrix}^T = \begin{bmatrix} 0 & 0 \end{bmatrix}^T$$

General eigen value problem is obtained as $K.A = \omega^2 M.A$. from which A, i.e., mode shapes and frequencies can be determined. Hence it can be written as,

$$\begin{vmatrix} 10001 - \lambda & -1 \\ -1 & 1 - \lambda \end{vmatrix} = 0 \quad (2.3)$$

which gives,

$$(10001 - \lambda)(1 - \lambda) - 1 = 0.$$

$$\lambda^2 - 10002\lambda + 10000 = 0.$$

$$\lambda = 1, 10001$$

where $\lambda = \omega^2$ and neglecting the negative sign, two values of ω can be obtained as $\omega_1 = 1$; $\omega_2 = 100.005$ and corresponding mode shapes are $A_{21} = 10000A_{11}$ and $A_{12} = -10000A_{22}$ where A_{ij} is the i^{th} mode at j^{th} frequency. By combining the above mode shapes U_1 and U_2 can be written as,

$$U_1 = A_{11} \sin(\omega_1 t + \psi_1) + A_{12} \sin(\omega_2 t + \psi_2)$$

$$U_2 = A_{21} \sin(\omega_1 t + \psi_1) + A_{22} \sin(\omega_2 t + \psi_2)$$

Unknowns are eliminated by the help of given initial conditions and hence the final response can be written as,

$$\begin{bmatrix} U_1 \\ U_2 \end{bmatrix} = \begin{bmatrix} 10^{-3} & 1 \\ 10 & -10^{-4} \end{bmatrix} \begin{bmatrix} \cos t \\ \cos(100.005t) \end{bmatrix} \quad (2.4)$$

Hence ^{response} ~~modes~~ of the system can be written as,

$$\begin{bmatrix} U_1 \\ U_2 \end{bmatrix} = \begin{bmatrix} 10^{-4} \\ 1 \end{bmatrix} 10 \cos t + \begin{bmatrix} 1 \\ -10^{-4} \end{bmatrix} \cos(100.005t) \quad (2.5)$$

TIME STEPPING METHODS

3.1 GENERAL

In time stepping methods, the equilibrium equations are integrated using a numerical step by step procedure. Numerical integration is based on two ideas. First, instead of trying to satisfy the equilibrium equation at any time t , it is aimed to satisfy only at discrete time intervals Δt apart. The second idea is that a variation of displacements, velocities and accelerations within each time interval Δt is assumed [1].

The various time stepping methods considered for the analysis of dynamic equilibrium equation are following:

- The Central difference method.
- The Houbolt method.
- The Newmark families of methods including
 - I. Constant average acceleration method.
 - II. Linear acceleration method.
- The Wilson theta method.

The derivation of each method is carried out for constant time interval Δt .

3.2 CENTRAL DIFFERENCE METHOD

Central difference method is a two step method. It is an explicit and only conditionally stable method [1,6]. In this method acceleration and velocity at time t can be written in terms of displacement at time t by the help of central difference. From figure 3.1 velocity in the middle of time interval can be written as,

$${}^{t-\Delta t/2} \circ \ddot{U} = \frac{{}^t U - {}^{t-\Delta t} U}{\Delta t} \quad (3.1)$$

$${}^{t+\Delta t/2} \circ \ddot{U} = \frac{{}^{t+\Delta t} U - {}^t U}{\Delta t} \quad (3.2)$$

Hence acceleration at time t can be written as,

$${}^t \circ \circ \ddot{U} = \frac{{}^{t+\Delta t/2} \circ \ddot{U} - {}^{t-\Delta t/2} \circ \ddot{U}}{\Delta t}$$

$${}^t \circ \circ \ddot{U} = \frac{({}^{t+\Delta t} U - 2 \cdot {}^t U + {}^{t-\Delta t} U)}{\Delta t^2} \quad (3.3)$$

Velocity at time t can be written as,

$${}^t \circ \dot{U} = \frac{({}^{t+\Delta t} U - {}^{t-\Delta t} U)}{2 \cdot \Delta t} \quad (3.4)$$

The displacement solution for time $t+\Delta t$ is obtained by considering equilibrium equation at the beginning of step, i.e., at time t,

$$M \cdot {}^t \circ \circ \ddot{U} + K \cdot {}^t U = {}^t R \quad (3.5)$$

Putting the value of acceleration at time t from equation (3.3) in equation (3.5) and after rearranging it is written as,

$$\left(\frac{M}{\Delta t^2} \right) {}^{t+\Delta t} U = {}^t R - \left(K - \frac{2}{\Delta t^2} M \right) {}^t U - \frac{M}{\Delta t^2} {}^{t-\Delta t} U \quad (3.6)$$

Thus the calculation of ${}^{t+\Delta t}U$ involves tU and ${}^{t-\Delta t}U$. ${}^{t-\Delta t}U$ can be obtained by the above relationships as given in equation (3.3) and equation (3.4).

$${}^{t-\Delta t}U = {}^tU - \overset{\circ}{U} \Delta t + \overset{\circ\circ}{U} \Delta t^2 / 2 \quad (3.7)$$

Thus from equation (3.6), displacement at next time step can be determined. Moreover acceleration and velocity at that time step can be determined by the aid of equation (3.3) and equation (3.4) respectively.

Hence the equilibrium is considered at time t to get the response of the system in the time step.

3.3 HOUBOLT METHOD

The Houbolt method is a three step method. In this method the acceleration and velocity at the end of the step is approximated in terms of displacements in the previous three steps. To derive the required relationship a cubic function of displacement is assumed passing through the four points as shown in figure 3.2. Hence displacement at any time is written as,

$$U(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3 \quad (3.8)$$

Where a_0, a_1, a_2, a_3 are constants to be determined.

velocity at any time t is written as,

$$\overset{\circ}{U}(t) = a_1 + 2a_2 t + 3a_3 t^2 \quad (3.9)$$

and acceleration at any time is written as,

$$\overset{\circ\circ}{U}(t) = 2a_2 + 6a_3 t \quad (3.10)$$

Considering the current displacement at $t = 0$, displacements in the next step and previous two steps can be written as

$$\begin{aligned}
{}^tU &= U(0) = a_0 \\
{}^{t+\Delta t}U &= U(\Delta t) = a_0 + a_1\Delta t + a_2\Delta t^2 + a_3\Delta t^3 \\
{}^{t-\Delta t}U &= U(-\Delta t) = a_0 - a_1\Delta t + a_2\Delta t^2 - a_3\Delta t^3 \\
{}^{t-2\Delta t}U &= U(-2\Delta t) = a_0 - 2a_1\Delta t + 4a_2\Delta t^2 - 8a_3\Delta t^3
\end{aligned}$$

The four constants can be determined by solving the above four equations, thus

$$\begin{aligned}
a_0 &= {}^tU \\
a_1 &= (2{}^{t+\Delta t}U + 3{}^tU - 6{}^{t-\Delta t}U + {}^{t-2\Delta t}U) / 6\Delta t \\
a_2 &= ({}^{t+\Delta t}U - 2{}^tU + {}^{t-\Delta t}U) / 2\Delta t^2 \\
a_3 &= ({}^{t+\Delta t}U - 3{}^tU + 3{}^{t-\Delta t}U - {}^{t-2\Delta t}U) / 6\Delta t^3
\end{aligned}$$

Hence acceleration and velocity can be written as,

$${}^{t+\Delta t} \overset{\circ\circ}{U} = (2{}^{t+\Delta t}U - 5{}^tU + 4{}^{t-\Delta t}U - {}^{t-2\Delta t}U) / \Delta t^2 \quad (3.11)$$

which follows,

$${}^{t+\Delta t} \overset{\circ}{U} = (11{}^{t+\Delta t}U - 18{}^tU + 9{}^{t-\Delta t}U - 2{}^{t-2\Delta t}U) / 6\Delta t \quad (3.12)$$

$${}^{t+\Delta t}U = (6\Delta t \overset{\circ}{U} + 18{}^tU - 9{}^{t-\Delta t}U + 2{}^{t-2\Delta t}U) / 11 \quad (3.13)$$

In order to obtain the displacement at time t , dynamic equilibrium equation at time $t+\Delta t$ is considered, i.e.,

$$M \overset{\circ\circ}{U} + K {}^{t+\Delta t}U = {}^{t+\Delta t}R \quad (3.14)$$

Substituting the values of acceleration as obtained above in equation (3.11) in equation (3.14) we get,

$$\left(\frac{2}{\Delta t^2}M + K\right) {}^{t+\Delta t}U = {}^{t+\Delta t}R + \frac{5}{\Delta t^2}M {}^tU - \frac{4}{\Delta t^2}M {}^{t-\Delta t}U + \frac{1}{\Delta t^2}M {}^{t-2\Delta t}U \quad (3.15)$$

Thus it is observed that to get the displacement at next step, in addition to displacement at present step the displacements at previous two steps are also required. In other words a special starting procedure is to be employed.

Features of the Houbolt method:

- It is an unconditionally stable method.
- It is an implicit method.
- A special starting procedure is required.
- It is a stiffly stable as well as A-stable method [6].

3.4 Newmark family of methods.

The Newmark methods are single step methods. The two methods considered under this family are Constant average acceleration method and linear acceleration method.

3.4.1 Constant average acceleration method

The acceleration is assumed to be constant and equal to the average of two accelerations in consecutive time steps. From figure 3.3 acceleration at any time can be written as,

$$\ddot{U}(t) = \frac{\ddot{U}_{t_{oo}} + \ddot{U}_{t+\Delta t_{oo}}}{2} \quad (3.16)$$

By successive integration velocity and displacement at any time t can be obtained respectively.

$$\dot{U}(t) = \dot{U}_{t_{oo}} + \frac{\ddot{U}_{t_{oo}} + \ddot{U}_{t+\Delta t_{oo}}}{2} t \quad U(t) = U_{t_{oo}} + \dot{U}_{t_{oo}} t + \frac{\ddot{U}_{t_{oo}} + \ddot{U}_{t+\Delta t_{oo}}}{4} t^2$$

Hence velocity and displacement at time $t+\Delta t$ can be written as,

$$\dot{U}_{t+\Delta t_{oo}} = \dot{U}_{t_{oo}} + \frac{\ddot{U}_{t_{oo}} + \ddot{U}_{t+\Delta t_{oo}}}{2} \Delta t \quad (3.17)$$

$${}^{t+\Delta t}U = {}^tU + {}^t\dot{U}\Delta t + \frac{{}^{t+\Delta t}\ddot{U} + {}^t\ddot{U}}{4}\Delta t^2 \quad (3.18)$$

3.4.2 Linear Acceleration method.

The linear variation of acceleration with respect to time is shown in figure 3.4. Thus acceleration at any time t can be written as,

$${}^t\ddot{U}(t) = {}^t\ddot{U} + \frac{{}^{t+\Delta t}\ddot{U} - {}^t\ddot{U}}{\Delta t}t \quad (3.19)$$

by integrating the above equation velocity can be written as,

$${}^t\dot{U}(t) = {}^t\dot{U} + {}^t\ddot{U}t + \frac{{}^{t+\Delta t}\ddot{U} - {}^t\ddot{U}}{2\Delta t}t^2 \quad (3.20)$$

Further integration gives displacement,

$$U(t) = {}^tU + {}^t\dot{U}t + \frac{{}^t\ddot{U}}{2}t^2 + \frac{{}^{t+\Delta t}\ddot{U} - {}^t\ddot{U}}{6\Delta t}t^3 \quad (3.21)$$

Hence velocity and displacement at time $t+\Delta t$ can be written as,

$${}^{t+\Delta t}\dot{U} = {}^t\dot{U} + \frac{{}^{t+\Delta t}\ddot{U} + {}^t\ddot{U}}{2}\Delta t \quad (3.22)$$

$${}^{t+\Delta t}U = {}^tU + {}^t\dot{U}\Delta t + \frac{{}^{t+\Delta t}\ddot{U} + {}^t\ddot{U}}{6}\Delta t^2 + \frac{{}^t\ddot{U}}{3}\Delta t^2 \quad (3.23)$$

In order to obtain the displacement at time t , dynamic equilibrium equation at time $t+\Delta t$ is considered, i.e.,

$$M^{t+\Delta t} \ddot{U} + K^{t+\Delta t} U = {}^{t+\Delta t}R \quad (3.24)$$

Putting the value of acceleration at time $t+\Delta t$ in equation (3.24) from the equation (3.16) and equation (3.19) displacement at time $t+\Delta t$ can be determined by constant average acceleration and Linear acceleration method respectively.

$$\left(\frac{4}{\Delta t^2} M + K\right)^{t+\Delta t} U = {}^{t+\Delta t}R + \frac{4}{\Delta t^2} M^t U + \frac{4}{\Delta t} M^t \dot{U} + M^t \ddot{U} \quad (3.25)$$

And

$$\left(\frac{6}{\Delta t^2} M + K\right)^{t+\Delta t} U = {}^{t+\Delta t}R + \frac{6}{\Delta t^2} M^t U + \frac{6}{\Delta t} M^t \dot{U} + 2M^t \ddot{U} \quad (3.26)$$

Thus by knowing the displacement, velocity and acceleration at time t , displacement can be determined at next time step $t+\Delta t$. Moreover first acceleration and then velocity in next time step can be determined.

Features of Newmark family of methods:

- Constant average acceleration method is an unconditionally stable method.
- Linear acceleration method is a conditionally stable method.
- Both are implicit methods.

3.5 The Wilson theta method

The Wilson theta method is essentially an extension of the Linear acceleration method, in which rather a linear variation of acceleration from time t to time $t+\theta\Delta t$ is assumed, where $\theta \geq 1.0$. When

$\theta = 1.0$, it reduces to Linear acceleration method. But for unconditional stability we need to use $\theta = 1.4$.

From figure 3.5 acceleration at any time t can be written as,

$$\ddot{U}(t) = \ddot{U} + \frac{\ddot{U} - \ddot{U}}{\theta \Delta t} t \quad (3.27)$$

By successive integration velocity and displacement at any time t can be obtained.

$$\dot{U}(t) = \dot{U} + \dot{U} t + \frac{\ddot{U} - \ddot{U}}{2\theta \Delta t} t^2$$

$$U(t) = \dot{U} t + \dot{U} \frac{t^2}{2} + \frac{\ddot{U} - \ddot{U}}{6\theta \Delta t} t^3$$

Hence velocity and displacement at time $t + \theta \Delta t$ can be written as,

$$\dot{U} = \dot{U} + \frac{\dot{U} + \dot{U}}{2} \theta \Delta t \quad (3.28)$$

$${}^{t+\theta\Delta t}U = {}^tU + \dot{U} \theta \Delta t + \frac{\ddot{U} + 2\ddot{U}}{6} \theta^2 \Delta t^2 \quad (3.29)$$

To obtain the solutions for the displacements, velocities and accelerations at time $t + \Delta t$, the equilibrium equation is considered at time $t + \theta \Delta t$.

$$M {}^{t+\theta\Delta t} \ddot{U} + K {}^{t+\theta\Delta t} U = {}^{t+\theta\Delta t} R \quad (3.30)$$

Where,

$${}^{t+\theta\Delta t} R = {}^t R + \theta ({}^{t+\Delta t} R - {}^t R)$$

Putting the value of acceleration at time $t+\theta\Delta t$ in equation (3.30) as derived in equation (3.29) displacement in the next time step can be obtained.

$$\left(\frac{6}{\theta^2\Delta t^2}M+K\right)^{t+\theta\Delta t}U=\frac{6}{\theta^2\Delta t^2}R+\frac{6}{\theta^2\Delta t^2}M^tU+\frac{6}{\theta\Delta t}M^t\dot{U}+2M^t\ddot{U} \quad (3.31)$$

Further velocity and acceleration in the next time step can be determined with the help of equation (3.28) and equation (3.29) respectively.

Features of Wilson theta method:

- It is an unconditionally stable method when $\theta \geq 1.37$.
- It is an implicit method.

STABILITY, ACCURACY AND DAMPING CHARACTERISTICS

4.1 STABILITY ANALYSIS [1,4]

Stability of an integration method means that the response in the higher modes i.e modes with large $\Delta t/T$ ratio must not be amplified by the method and thus render worthless any accuracy in the integration of the lower mode response. It also means that any initial condition at the beginning of the time step given by errors in the displacements, velocities and accelerations which may be due to round off in the computer, do not grow in the integration. Stability is ensured if the time step is small enough to integrate accurately the response in the highest frequency component of the system. In the derivation of various time stepping methods, behavior of system is considered at discrete time interval. Then for specific time stepping method considered, we aim to establish the following recursive relationship,

$${}^{t+\Delta t}X = A {}^tX + L {}^{t+v}r \quad (4.1)$$

where ${}^{t+\Delta t}X$ and tX are vectors storing the solution quantities, i.e., displacements, velocities and accelerations. ${}^{t+v}r$ is the load at time $t+v$. v may be 0, Δt or $\theta\Delta t$ for the central difference method, the Houbolt method or the Newmark methods and Wilson theta method respectively. The matrix A and vector L are the amplification matrix and load ^{operator} vector, respectively.

4.2 DERIVATION OF MATRICES AND VECTORS CORRESPONDING TO DIFFERENT TIME STEPPING METHODS[1,4]

Variables to be considered in the stability analysis of the time stepping methods are only Δt and ω and not all elements of the stiffness and mass matrices. Furthermore, because all n equations are similar we only need to study the integration of one typical row, which may be written as

$${}^{\infty}x + \omega^2 x = r \quad (4.2)$$

4.2.1 The Central Difference Method:

In the Central difference method acceleration and velocity are approximated at time t . Also equilibrium equation is considered at time t .

$${}^{\infty}x + \omega^2 x = r \quad (4.3)$$

$$x = \frac{1}{\Delta t^2} ({}^{t-\Delta t}x - 2{}^t x + {}^{t+\Delta t}x) \quad (4.4)$$

$$x = \frac{1}{2\Delta t} ({}^{t-\Delta t}x + {}^{t+\Delta t}x) \quad (4.5)$$

substituting equation (4.4) in equation (4.3) and solving for ${}^{t+\Delta t}x$, we obtain

$${}^{t+\Delta t}x = (2 - \omega^2 \Delta t^2) {}^t x - {}^{t-\Delta t}x + \Delta t^2 r$$

The solution can be written in the form

$$\begin{bmatrix} {}^{t+\Delta t}x \\ {}^t x \end{bmatrix} = A \begin{bmatrix} {}^t x \\ {}^{t-\Delta t}x \end{bmatrix} + L r \quad (4.6)$$

where

$$A = \begin{bmatrix} 2 - \omega^2 \Delta t^2 & -1 \\ 1 & 0 \end{bmatrix} \text{ and } L = \begin{bmatrix} \Delta t^2 \\ 0 \end{bmatrix}$$

4.2.2 The Houbolt method

In the Houbolt method the equilibrium equation is considered at time $t+\Delta t$. The acceleration and velocity at time $t+\Delta t$ can be written as derived before.

$${}^{t+\Delta t} \ddot{x} + \omega^2 {}^{t+\Delta t} x = {}^{t+\Delta t} \Gamma \quad (4.7)$$

$${}^{t+\Delta t} \dot{x} = \frac{1}{\Delta t^2} (2 {}^{t+\Delta t} x - 5x + 4 {}^{t-\Delta t} x - {}^{t-2\Delta t} x) \quad (4.8)$$

$${}^{t+\Delta t} x = \frac{1}{6\Delta t} (1 {}^{t+\Delta t} x - 18x + 9 {}^{t-\Delta t} x - 2 {}^{t-2\Delta t} x) \quad (4.9)$$

by rearranging the above equations it can be written in the matrix form as,

$$\begin{bmatrix} {}^{t+\Delta t} x \\ x \\ {}^{t-\Delta t} x \end{bmatrix} = A \begin{bmatrix} x \\ {}^{t-\Delta t} x \\ {}^{t-2\Delta t} x \end{bmatrix} + L {}^{t+\Delta t} \Gamma \quad (4.10)$$

where,

$$A = \begin{bmatrix} \frac{5\beta}{\omega^2 \Delta t^2} & -\frac{4\beta}{\omega^2 \Delta t^2} & \frac{\beta}{\omega^2 \Delta t^2} \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \quad \text{where } \beta = \left(1 + \frac{2}{\omega^2 \Delta t^2}\right)^{-1} \text{ and } L = \begin{bmatrix} \beta \\ 0 \\ 0 \end{bmatrix}$$

4.2.3 The Wilson theta method

The basic assumption in the Wilson theta method is that the acceleration varies linearly over the time interval from t to $t+\theta\Delta t$, where $\theta \geq 1$. Let τ denote the increase in time from t , where $0 \leq \tau \leq \theta\Delta t$ then acceleration, velocity and displacement in this time interval can be written as,

$${}^{t+\tau}x = {}^t x + ({}^t x - {}^{t+\Delta t}x) \frac{\tau}{\Delta t} \quad (4.11)$$

$${}^{t+\tau}x = {}^t x + {}^t x \tau + ({}^t x - {}^{t+\Delta t}x) \frac{\tau^2}{2\Delta t} \quad (4.12)$$

$${}^{t+\tau}x = {}^t x + {}^t x \tau + \frac{1}{2} {}^t x \tau^2 + ({}^t x - {}^{t+\Delta t}x) \frac{\tau^3}{6\Delta t} \quad (4.13)$$

At time $t + \Delta t$ we have

$${}^{t+\Delta t}x = {}^t x + ({}^t x + {}^{t+\Delta t}x) \frac{\Delta t}{2} \quad (4.14)$$

$${}^{t+\Delta t}x = {}^t x + {}^t x \Delta t + ({}^t x + 2{}^{t+\Delta t}x) \frac{\Delta t^2}{6} \quad (4.15)$$

In the Wilson theta method the equilibrium is considered at time $t + \theta \Delta t$,

$${}^{t+\theta \Delta t}x + \omega^2 {}^{t+\theta \Delta t}x = {}^{t+\theta \Delta t}r \quad (4.16)$$

By rearranging the above equations the following relationship can be established as,

$$\begin{bmatrix} {}^{t+\Delta t}x \\ {}^{t+\Delta t}x \\ {}^{t+\Delta t}x \\ {}^{t+\Delta t}x \end{bmatrix} = A \begin{bmatrix} {}^t x \\ {}^t x \\ {}^t x \\ {}^t x \end{bmatrix} + L {}^{t+\theta \Delta t}r \quad (4.17)$$

where,

$$A = \begin{bmatrix} (1 - \frac{\beta \theta^2}{3} - \frac{1}{\theta}) & -\frac{\beta \theta}{\Delta t} & -\frac{\beta}{\Delta t^2} \\ \Delta t (1 - \frac{1}{2\theta} - \frac{\beta \theta^2}{6}) & 1 - \frac{\beta \theta}{2} & -\frac{\beta}{2\Delta t} \\ \Delta t^2 (\frac{1}{2} - \frac{1}{6\theta} - \frac{\beta \theta^2}{18}) & \Delta t (1 - \frac{\beta \theta}{6}) & 1 - \frac{\beta}{6} \end{bmatrix} \quad \text{and} \quad L = \begin{bmatrix} \frac{\beta}{\omega^2 \Delta t^2} \\ \frac{\beta}{2\omega^2 \Delta t} \\ \frac{\beta}{6\omega^2} \end{bmatrix}$$

4.2.4 The Newmark method

In the Newmark method the equilibrium is considered at time $t+\Delta t$.

$$M \ddot{U} + K U = R \quad (4.18)$$

The expression for velocity and displacement at time $t+\Delta t$ can be written as,

$$\dot{x} = \dot{x}_0 + \left[(1-\delta) \ddot{x}_0 + \delta \ddot{x}_{t+\Delta t} \right] \Delta t \quad (4.19)$$

$$x = x_0 + \dot{x}_0 \Delta t + \left[\left(\frac{1}{2} - \alpha \right) \ddot{x}_0 + \alpha \ddot{x}_{t+\Delta t} \right] \Delta t^2 \quad (4.20)$$

By rearranging the above equations in the matrix form,

$$\begin{bmatrix} \ddot{x}_{t+\Delta t} \\ \dot{x}_{t+\Delta t} \\ x_{t+\Delta t} \end{bmatrix} = A \begin{bmatrix} \ddot{x}_0 \\ \dot{x}_0 \\ x_0 \end{bmatrix} + L \ddot{x}_{t+\Delta t} \quad (4.21)$$

where,

$$A = \begin{bmatrix} -\left(\frac{1}{2} - \alpha\right)\beta & \frac{-\beta}{\Delta t} & \frac{-\beta}{\Delta t^2} \\ \Delta t \left[1 - \delta - \left(\frac{1}{2} - \alpha\right)\delta\beta \right] & 1 - \beta\delta & \frac{-\beta\delta}{\Delta t} \\ \Delta t^2 \left[\frac{1}{2} - \alpha - \left(\frac{1}{2} - \alpha\right)\alpha\beta \right] & \Delta t(1 - \alpha\beta) & 1 - \alpha\beta \end{bmatrix} \quad \text{and } L = \begin{bmatrix} \frac{\beta}{\omega^2 \Delta t^2} \\ \frac{\beta\delta}{\omega^2 \Delta t} \\ \frac{\alpha\beta}{\omega^2} \end{bmatrix}$$

$$\text{where } \beta = \left(\frac{1}{\omega^2 \Delta t^2} + \alpha \right)^{-1}$$

The stability of an integration method is therefore determined by examining the behavior of the numerical solution for arbitrary initial conditions [1].

Therefore we consider the integration of equation (4.1) when no load is present, i.e. $r=0$. Hence

$${}^{t+\Delta t}X = A {}^tX \quad (4.22)$$

the stability of the system can be explained by the eigen value problem as $|A - \lambda I| = 0$. Let $\rho(A)$ be the spectral radius of matrix A which is defined as,

$$\rho(A) = \max_{i=1,2,\dots} |\lambda_i| \quad (4.23)$$

Then stability criterion is that

- If all eigen values are distinct, we must have $\rho(A) \leq 1$, whereas
- If A contains multiple eigen values, we require that all such eigen values in absolute magnitude be smaller than 1.

Thus the stability of time stepping methods depends only on the time ratio $\Delta t/T$ and the integration parameters used. Therefore, for given $\Delta t/T$, it is possible in the Wilson theta method and in the Newmark method to vary the parameters θ and α, δ , respectively to obtain stability characteristics. We will see later that that it is also true for accuracy analysis. Now the stability analysis of the Central difference method is considered. The eigen value problem to be solved is

$$|A - \lambda I| = 0.$$

Where

$$A = \begin{bmatrix} 2 - \omega^2 \Delta t^2 & -1 \\ 1 & 0 \end{bmatrix}$$

by solving the two values of λ are obtained as,

$$\lambda_{1,2} = \frac{2 - \omega^2 \Delta t^2}{2} \pm \sqrt{\frac{(2 - \omega^2 \Delta t^2)^2}{4} - 1} \quad (4.24)$$

for stability we need that the absolute value of λ_1 and λ_2 be smaller than or equal to 1 and this gives the condition $\Delta t/T \leq 1/\pi$.

Hence the central difference method is stable provided that

$\Delta t \leq \Delta t_{cr}$ where $\Delta t_{cr} = T_n/\Pi$. By the same procedure as employed above the Wilson theta, Newmark and Houbolt methods can be analyzed for stability using the corresponding approximation operators. It is noted that central difference method is only conditionally stable and the other three methods are unconditionally stable. Figure 4.1 shows the plot of spectral radius of amplification matrix of different methods with respect to ratio of time step. Moreover in order to obtain the optimum value of θ for the Wilson theta method, the variation of the spectral radius of approximation operator with respect to θ is plotted as in figure 4.2. It is seen that unconditional stability is obtained when $\theta \geq 1.37$. Considering the Newmark method, it is unconditionally stable when $\delta \geq 0.5$ and $\alpha \geq 0.25(\delta+0.5)^2$

4.3 ACCURACY ANALYSIS [1,4,6]

The choice of particular time stepping method depend upon the cost of analysis which in turn depends upon the number of time steps required in the integration. In the case of conditionally stable method like the Central difference method where the time step is determined by the critical time step, not much choice is available. However, using an unconditionally stable method, the step has to be chosen to yield an accurate and effective solution.

Let us consider for a simple accuracy analysis the solution of the initial value problem defined by

$$\ddot{x} + \omega^2 x = 0 \quad (4.25)$$

$$\text{and } x^0 = 1.0; \quad \dot{x}^0 = 0.0; \quad \dot{x}^{\infty} = -\omega^2$$

The above equation has the exact solution as $x = \cos \omega t$. The Newmark and Wilson theta methods can be used directly however in the Houbolt method, the initial conditions are defined only by initial displacements, and in the following study the exact displacement values for $\Delta t x$ and $2\Delta t x$ are obtained using the solution of $x = \cos \omega t$.

The numerical solution of equation (4.25) using the different integration methods show that the errors in the integration can be measured in terms of period elongation and amplitude decay. Figure 4.3 shows the percentage period elongation and amplitude decay as a function of $\Delta t/T$. The curves in figure 4.3 shows that the methods are accurate when $\Delta t/T$ is smaller than about 0.01. However when $\Delta t/T$ is larger, the various time stepping methods exhibit quite different characteristics. For a given $\Delta t/T$, the Wilson theta method with $\theta = 1.4$ introduces less amplitude decay and period elongation than the Houbolt method, and the Newmark's constant average acceleration method introduces only period elongation and no amplitude decay.

The response calculated by the four methods in fundamental and higher modes reveals that the amplitude decay caused by the numerical integration effectively filters the high mode response out of the solution in the Wilson theta and Houbolt method. Whereas, when constant average acceleration is employed the frequency response is retained in the solution as it does not introduce amplitude decay. In order to obtain amplitude decay using the Newmark method, it is recommended that $\delta > 0.5$ and correspondingly $\alpha = 0.25(\delta+0.5)^2$ as Damped newmark's response spectra is obtained by $\delta = 0.6$ and $\alpha = 0.3025$.

4.4 NUMERICAL DISSIPATION [4,6]

In many structural dynamics application only low mode response is of interest. For these cases the use of implicit unconditionally stable algorithms are generally preferred over conditionally stable algorithm. The reason is that, in conditionally stable algorithm the size of time step employed is inversely proportional to the highest frequency of the discrete system. In practice this is a severe limitation as accuracy in the lower mode can be attained with the time steps, which are very large compared with the period of the highest mode. In unconditionally stable, when only low mode response is of interest it is often advantageous for an

algorithm to possess some form of numerical damping to damp out any participation of higher modes. Algorithm commonly used in structural dynamics which possess these properties are the Houbolt method, the Wilson theta method and the Newmark family of methods restricted to parameters with value of $\delta > 1/2$ and $\alpha \geq 0.25 \delta (\delta + 0.5)^2$.

The Newmark family of method allows the amount of dissipation to be continuously controlled by a parameter other than the time step. For example set $\alpha = 0.25(\delta + 0.5)^2$ and $\delta > 1/2$, then the amount of dissipation, for a fixed time step is increased by increasing δ . On the other hand dissipative property of this family is considered to be inferior, since lower modes are affected too strongly. In Wilson theta method θ must be selected greater than or equal to 1.37 to maintain unconditional stability. It is recommended that $\theta = 1.4$ be employed as further increasing θ reduces accuracy and further increases dissipation. Houbolt method is even more dissipative than Wilson theta method. Since it seemed that the commonly used unconditionally stable, dissipative algorithm of structural dynamics all possessed some drawbacks, a research work was undertaken to see if an improved one step method could be constructed. In the Newmark family a new form of dissipation called α -dissipation, was introduced. [4,6], which could not be included in this dissertation.

A close study of the plot in figure 4.1 reveals that no dissipation is present in the Constant average acceleration method and Houbolt method possesses the strongest dissipation

4.5 LOGARITHMIC DAMPING

It is noted that the logarithmic decrement which is given by

$$\bar{\delta} = \ln \left[\frac{U(t_n)}{U(t_{n+m})} \right] = \left(\frac{2m\Gamma\xi}{\sqrt{1-\xi^2}} \right) \quad (4.26)$$

is commonly used measures of algorithmic dissipation. m indicates the number of ~~steps~~^{cycles}. Following table shows logarithmic decrement with different values of damping ratios and at different number of ~~time steps~~^{cycles}.

TABLE 4.1

No. of steps cycles → ξ↓	1	10	100	1000
0.001	0.9937	0.9391	0.5335	1.9×10^{-3}
0.01	0.9391	0.5335	1.9×10^{-3}	5.1×10^{-28}
0.02	0.8819	0.2845	3.5×10^{-6}	2.6×10^{-55}
0.03	0.8281	0.1517	6.5×10^{-9}	1.3×10^{-82}
0.04	0.7776	0.0808	1.2×10^{-11}	0
0.05	0.7301	0.0430	2.2×10^{-14}	0
0.1	0.5318	1.8×10^{-3}	3.8×10^{-28}	0
0.2	0.2773	2.7×10^{-6}	2.0×10^{-56}	0

The above quantities in the table is plotted in figure 4.5. The study of Table 4.1 reveals that logarithmic decrement tends to zero even in ten steps having $\xi = 0.2$ and in hundred steps having $\xi = 0.02$, which is not obtained by any algorithms, not even by the Houbolt method which possesses highest damping among the methods discussed.

STUDY OF RESPONSE HISTORY

5.1 General

As it is seen in mode superposition method that the response contains low frequency modes as well as high frequency modes. In the problem that we have considered it can be written as,

$$\begin{bmatrix} U_1 \\ U_2 \end{bmatrix} = \begin{bmatrix} 10^{-4} \\ 1 \end{bmatrix} 10 \cos t + \begin{bmatrix} 1 \\ -10^{-4} \end{bmatrix} \cos 100.005 t \quad (5.1)$$

the first part contains low frequency and second part high frequency component. In other words U_1 represents higher modes and U_2 represents lower modes. It represents the sinusoidal wave with maximum amplitude as 1 for U_1 in high frequency component and 10 for U_2 in low frequency component. From the above equation velocity can be written as,

$$\begin{bmatrix} \dot{U}_1 \\ \dot{U}_2 \end{bmatrix} = -\begin{bmatrix} 10^{-4} \\ 1 \end{bmatrix} 10 \sin t - \begin{bmatrix} 1 \\ -10^{-4} \end{bmatrix} 100.005 \sin 100.005 t \quad (5.2)$$

it represents the sinusoidal wave with maximum amplitude multiplied by 100.005 and 1 respectively. By extending the same operation it is obvious that acceleration response is further multiplied by 100.005 and 1 respectively. With this background now we try to analyze the response history obtained by all methods individually.

5.2 Central difference method.

It shows the beating phenomenon for response in the high frequency region. ^(Figures 5.1, 5.2 and 5.3) Also response in the low frequency region is not able to complete even one cycle. ^(Figures 5.4, 5.5 and 5.6) It is due to the smaller time step considered as it is conditionally stable method. The time step considered in this case is 0.01999 in place of conventional time step as 0.314159. Moreover in high

frequency response peak is obtained after more than 20 cycles which is due to elongation of time period.

5.3 Trapezoidal method.

It also shows the beating phenomenon in the high frequency region, λ but in the low frequency region exact response is obtained with very small damping in the higher time steps, λ . As maximum amplitude of the response is concerned, it gives the exact figure as discussed above. In this case also period elongation is observed. Wrinkles occur in the acceleration response because high frequency component of acceleration response is not damped out, which adversely affects the acceleration response in low frequency region. But it is not going to affect the displacement and velocity response as smooth curves are obtained. This phenomenon can be better understood by the response expression as written in equation (5.1).

$$U_2 = 10 \cos t \text{ (displacement response in low frequency region)}$$

$$U_2 = 10^{-4} \cos 100.005t \text{ (contribution from high frequency region)}$$

$$\dot{U}_2 = 10 \sin t \text{ (velocity response in low frequency region)}$$

$$\dot{U}_2 = 10^{-2} \sin 100.005t \text{ (contribution from high frequency region)}$$

Obviously contribution from high frequency region is negligible.

But

$$\ddot{U}_2 = 10 \cos t \text{ (acceleration response in low frequency region)}$$

$$\ddot{U}_2 = \cos 100.005t \text{ (contribution from high frequency region)}$$

It shows that contribution from high frequency component is one-tenth of low frequency component due to which acceleration response in low frequency region is affected.

5.4 Damped Newmark method.

It is able to damp out responses in both frequency regions, ^(Figures 5.13, 5.14 and 5.15) Although damping is small in low frequency region, ^(Figures 5.16, 5.17 and 5.18) it completely damps out responses in high frequency region in higher time steps. Moreover in the lower time steps also, damping is prevalent particularly in high frequency region. Smooth curves are obtained for all three quantities in low frequency region, except for initial wrinkles in acceleration response, because as it is clear from the response in high frequency region that although acceleration response is damped out in higher time steps, it is present in lower time steps.

5.5 Wilson theta method

Although it is able to damp out responses rapidly as compared to previous method in the high frequency region, ^(Figures 5.25, 5.26 and 5.27) but the responses amplitude in the lower time steps exceeds the value that discussed above. The responses in the low frequency region is okay in a sense that exact figure is obtained with smoothness except for initial wrinkles in the acceleration response and very small damping in the higher time steps, ^(Figures 5.28, 5.29 and 5.30)

5.6 Houbolt method.

The damping of responses in high frequency region is even fast as compared to previous method, or in other words it is fastest in all the methods discussed, ^(Figures 5.19, 5.20 and 5.21) which is desirable and moreover damping is small in low frequency region, ^(Figures 5.22, 5.23 and 5.24) Since acceleration response is totally damped out in just few steps, smooth curves are obtained for all quantities in low frequency region.

CONCLUSIONS

The study of various time stepping methods reveals that the Central difference method and Newmark's linear acceleration method are conditionally stable methods. This is a stringent condition leaving no choice in selecting the time steps. Naturally unconditionally stable methods are our choice. Among other unconditionally methods discussed the best method identified is the Houbolt method. The aim is to obtain the exact response in lower frequency region which can be obtained by filtering the high frequency response. It is evident from the response spectra obtained by the Houbolt method that the smooth curves with very small damping is observed in the low frequency region. It is due to the fact that responses of the quantities in high frequency region is damped out in just few steps, which renders the desired response unaffected.

The superiority of the Houbolt method is further supported by studying the damping phenomenon of different time stepping methods in the plot of spectral radius vs ratio of time steps. The spectral radius for the Houbolt method goes to the minimum value as compared to the other methods.

As we compare the plot of spectral radius vs. ratio of time steps with the plot of displacement decrement vs. damping values and number of time steps, the Houbolt method although having the largest damping properties, doesn't damp the response in the high frequency region as it should damp as seen in the plot of logarithmic damping with comparable damping value and comparable number of time steps.

SCOPE FOR FURTHER STUDY

In present study we investigated undamped systems only. However, a more descriptive behavior of system can be obtained by considering the damping in the system. This dissertation would be helpful in generalization for damped cases also.

REFERENCES

1. BATHE K. JURGEN, "Finite Element Procedures", Prentice Hall of India Pvt. Ltd., N.Delhi, 1996.
2. CHOPRA A. K., "Dynamics of Structure", Prentice Hall of India Pvt. Ltd., N.Delhi, 1996.
3. CLOUGH R. W. and PENZIEN, J., "Dynamics of structures", Mcgraw Hill Company, Singapore, 1993.
4. HILBER H. M., HUGHES T. J. R., and TAYLOR R. L.. "Improved Numerical Dissipation For Time Integration Algorithms in Structural Dynamics", Earthquake Engineering and Structural Dynamics, 5(1977), 283-292.
5. HILBER H. M. and HUGHES T. J. R., "Collocation, Dissipation and Overshoot for Time Integration schemes in Structural Dynamics", Earthquake Engineering and Structural Dynamics, 6(1978), 99-118.
6. HUGHES THOMAS J. R., "The Finite Element Method ", Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1987.
7. KRIEG R. D., "Unconditional Stability in Numerical Time Integration Methods", Journal of Applied Mechanics, 40(1973), 417-421.
8. LAFORE ROBERT "Object Oriented Programming in Turbo C++", Galgotia Publications Pvt. Ltd., N.Delhi, 1998.
9. PRESS WILLIAM H., TEUKOLSKY SAUL A., VETTERLING WILLIAM T. and FLANNERY BRIAN P., "Numerical Recipes in C", Cambridge University Press, 1992.
10. TSE FRANCIS S, MORSE IVAN E. AND HINKLE ROLLAND T., "Mechanical Vibrations" Prentice Hall of India Pvt. Ltd., N.Delhi, 1968.
11. ZIENKIEWICZ O. C., WOOD, W. L., HINE, N. W. AND TAYLOR, R. L., "A Unified Set of Single Step Algorithms", International Journal For Numerical Methods in Engineering, 20(1984), 1529-1552.

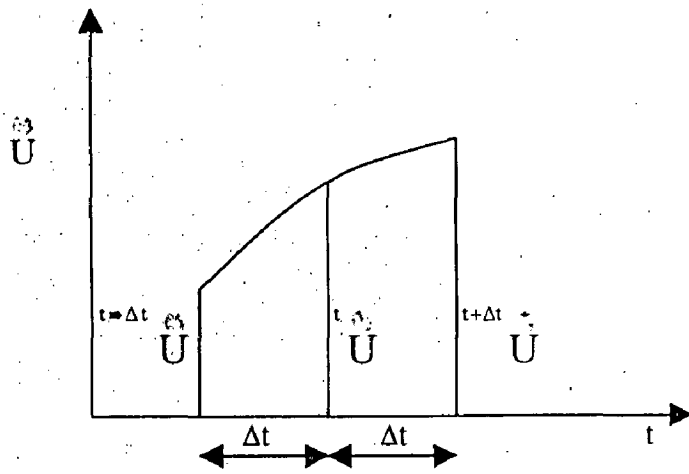


Figure 3.1 Displacement vs Time [Central Difference method]

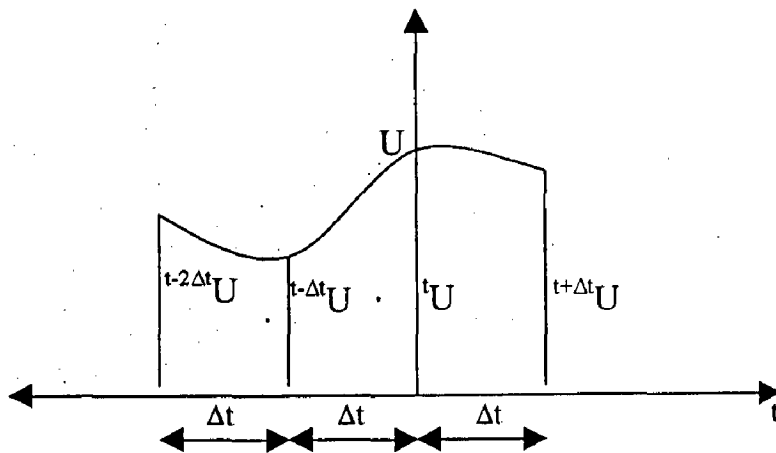


Figure 3.2 Displacement vs Time [Houbolt method]

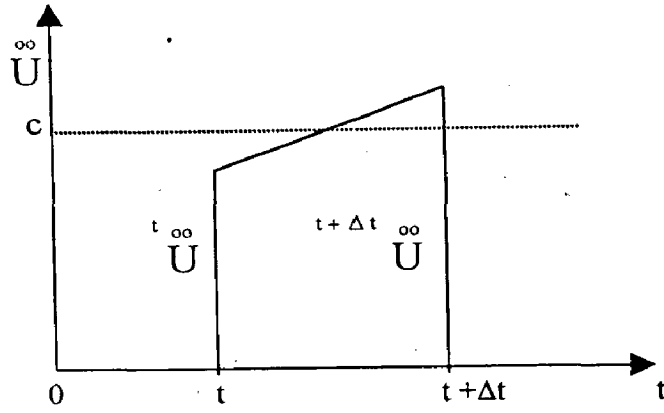


Figure 3.3 Acceleration vs Time [Constant Average Acceleration method]

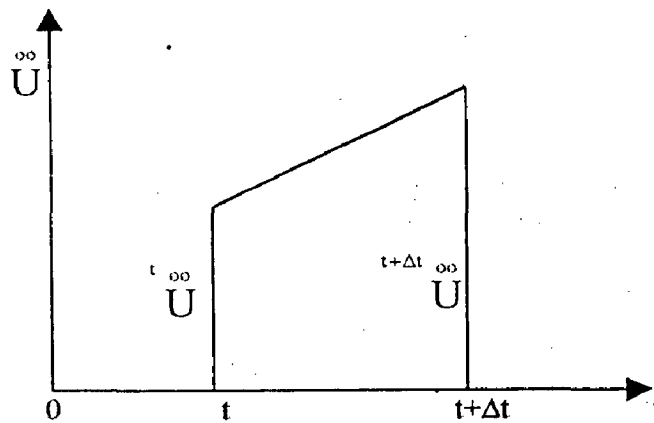


Figure 3.4 Acceleration vs Time [Linear Acceleration method]

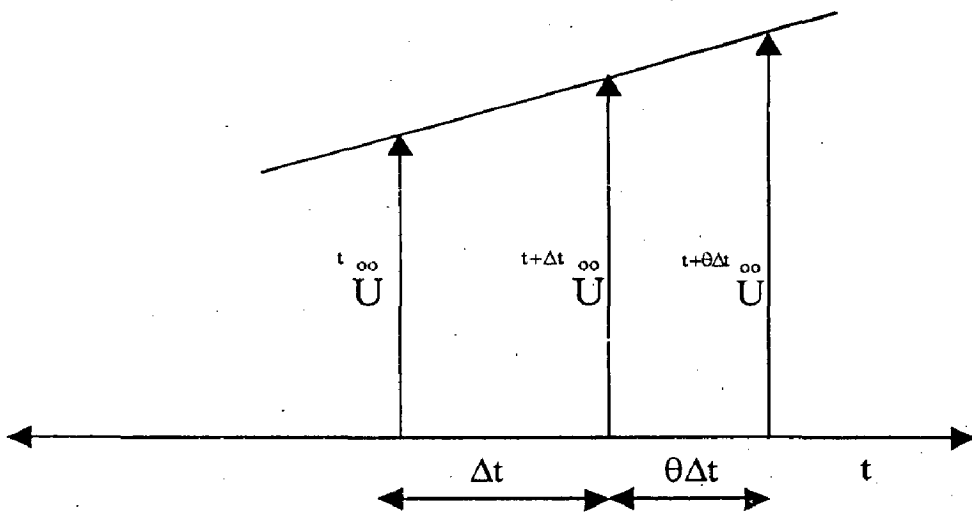


Figure 3.5 Acceleration vs Time [Wilson θ method]

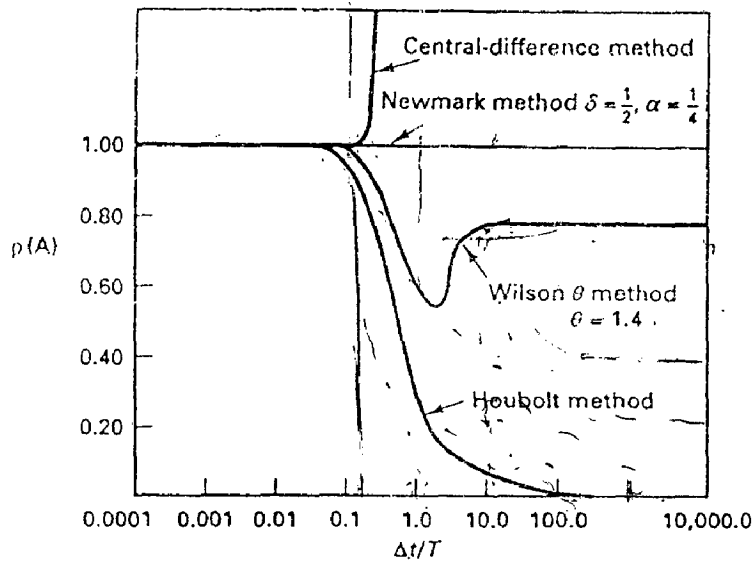


Figure 4.1 Spectral radii of amplification matrix, case $\xi = 0.0$

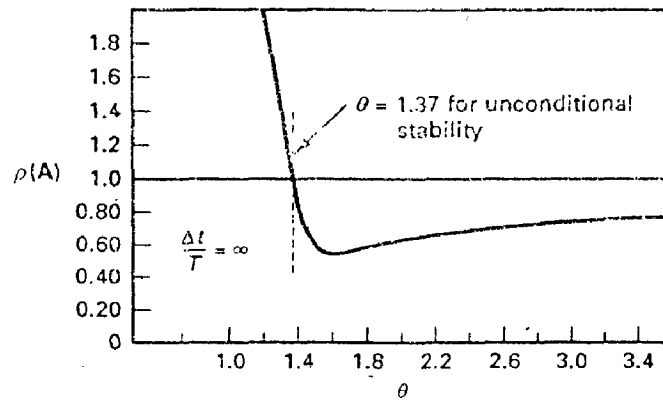


Figure 4.2 Spectral radius $\rho(A)$ as a function of θ in the Wilson θ method

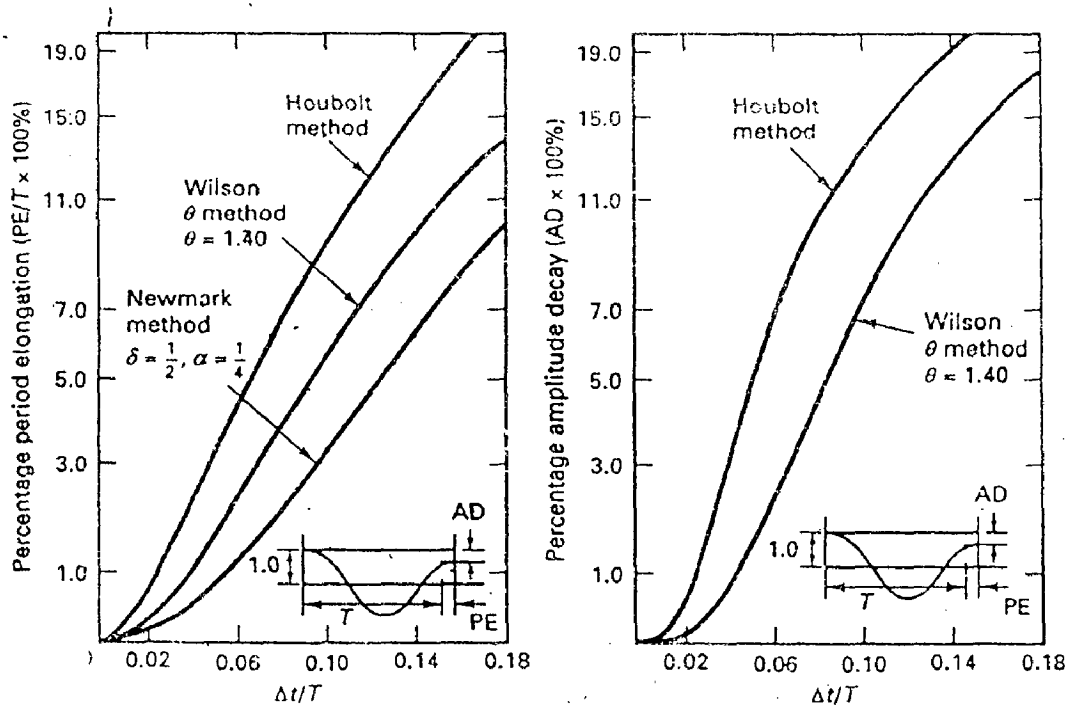
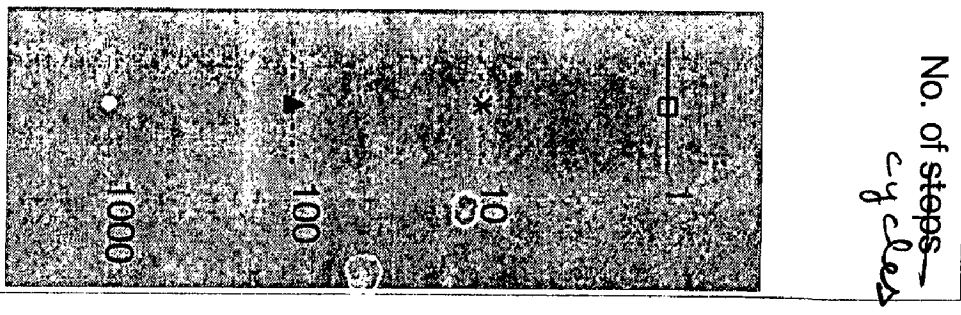
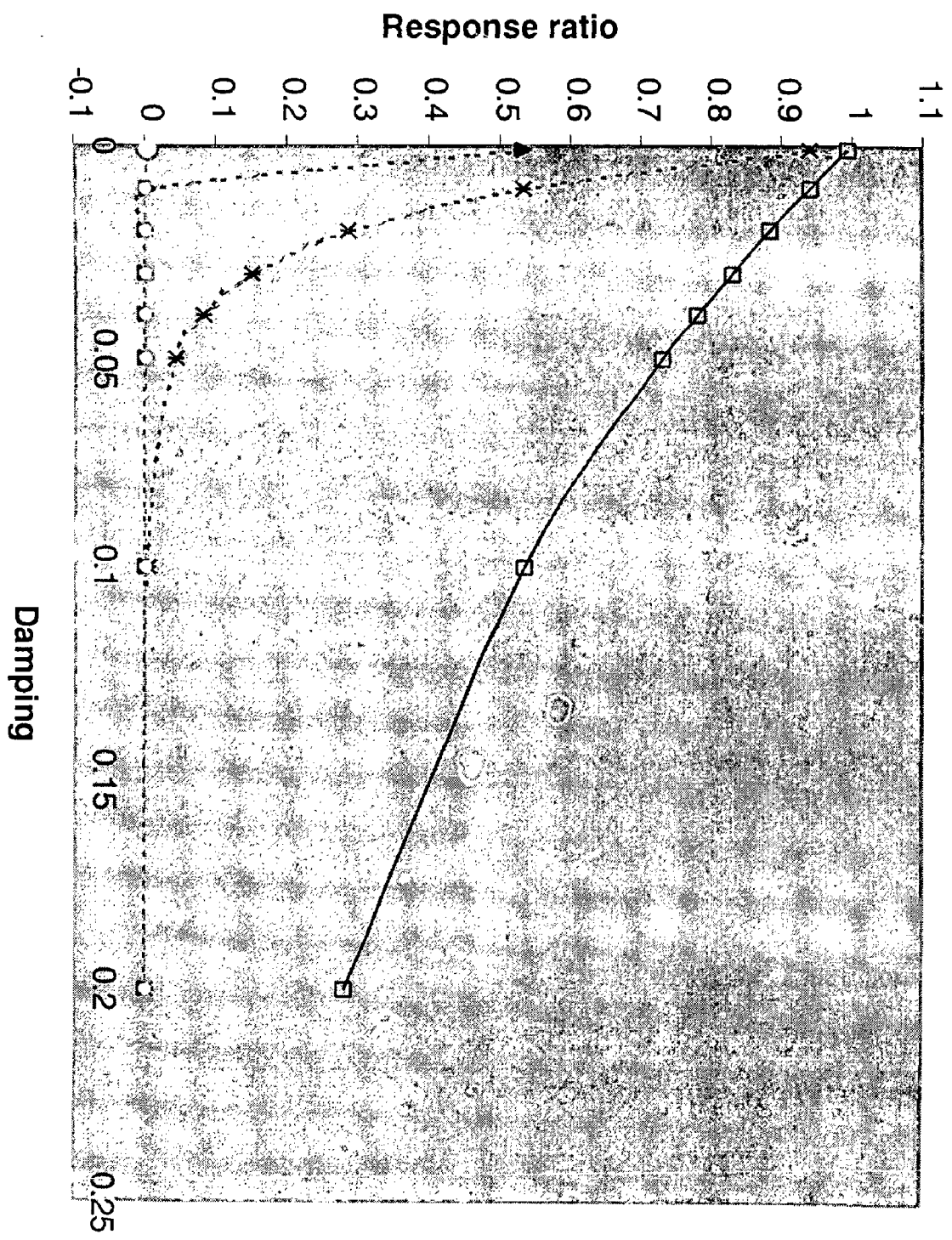


Figure 4.3 Percentage period elongations and amplitude decay

Figure 4.4 PLOT OF RESPONSE vs LOGARITHMIC DAMPING



TIME HISTORY PLOT FOR DISPLACEMENT, VELOCITY AND ACCELERATION

CENTRAL DIFFERENCE METHOD

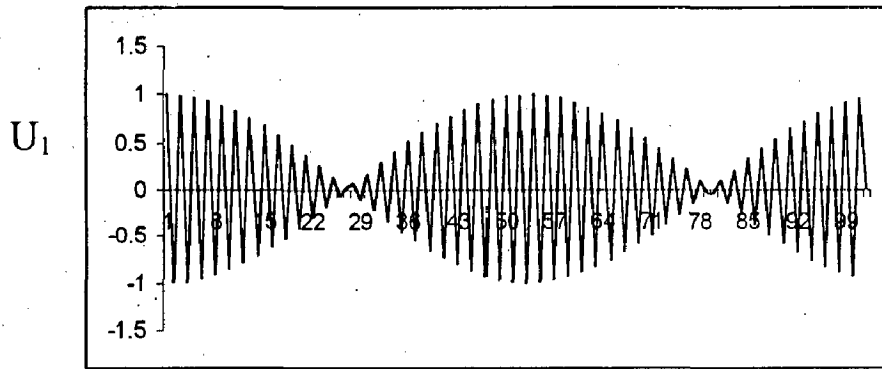


Figure 5.1 Displacement vs. time steps

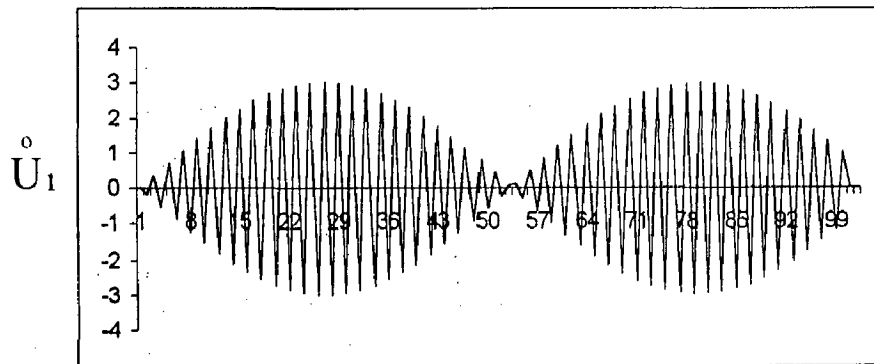


Figure 5.2 Velocity vs. time steps

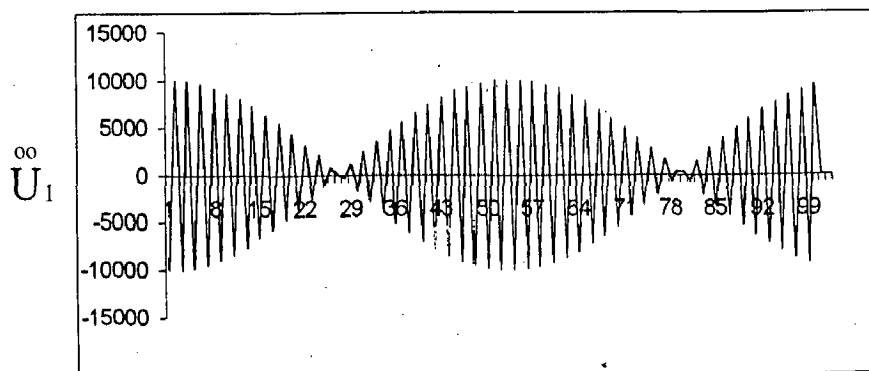


Figure 5.3 Acceleration vs. time steps

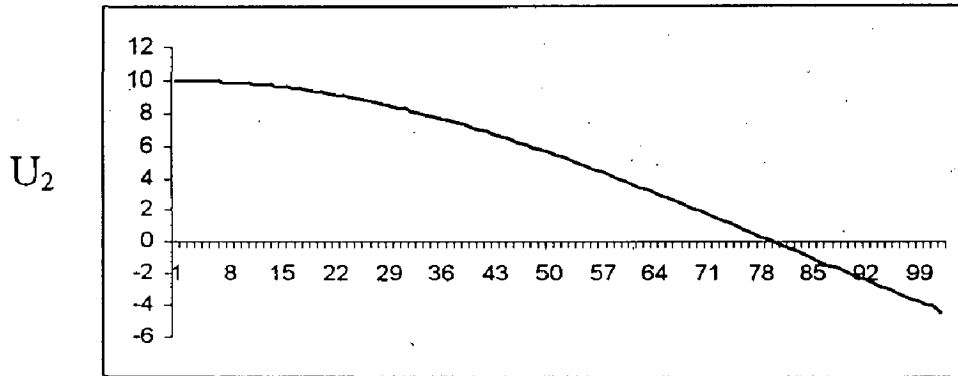


Figure 5.4 Displacement vs. time steps

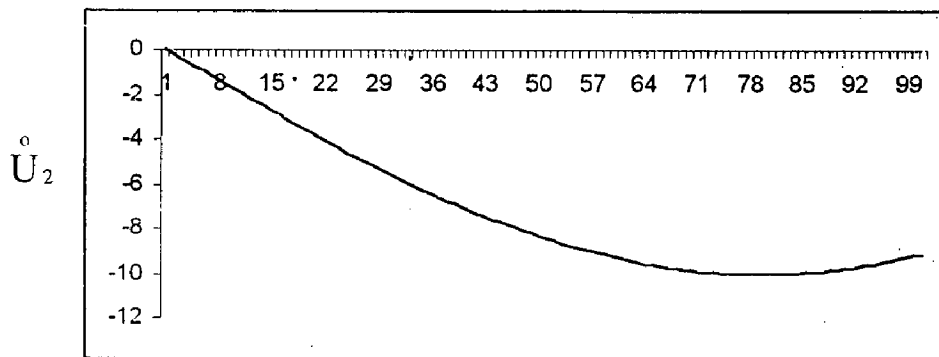


Figure 5.5 Velocity vs. time steps

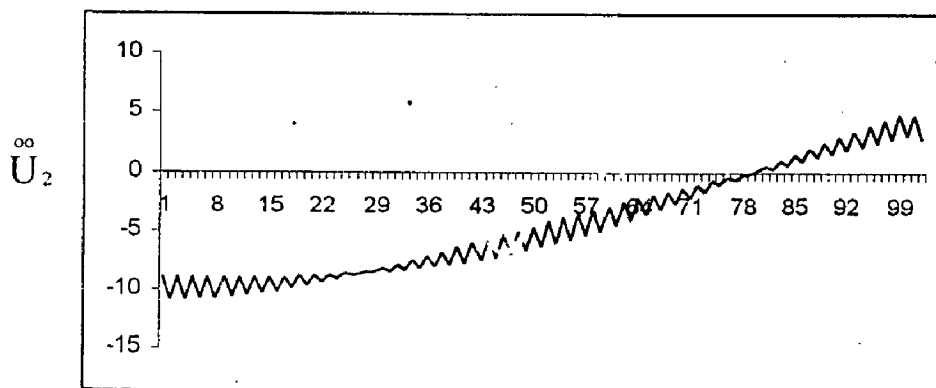


Figure 5.6 Acceleration vs. time steps

TRAPEZOIDAL METHOD

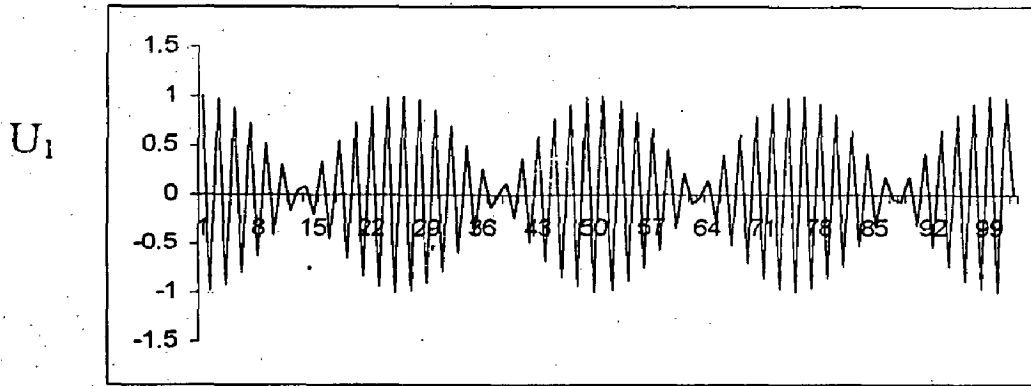


Figure 5.7 Displacement vs. time steps

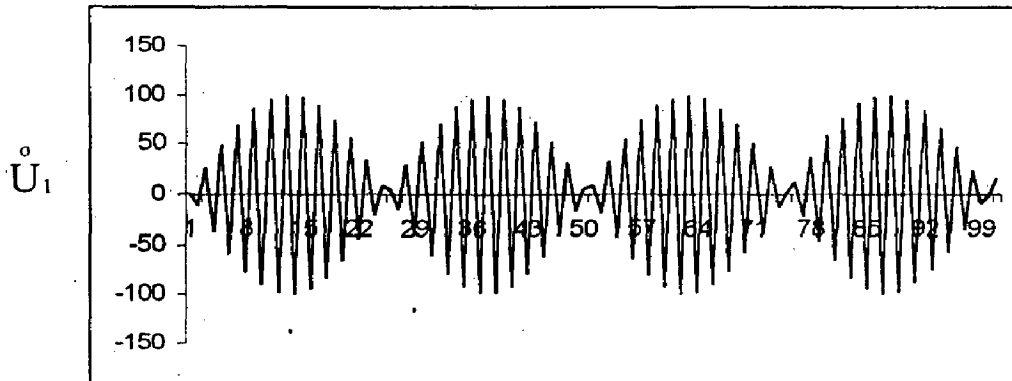


Figure 5.8 Velocity vs. time steps

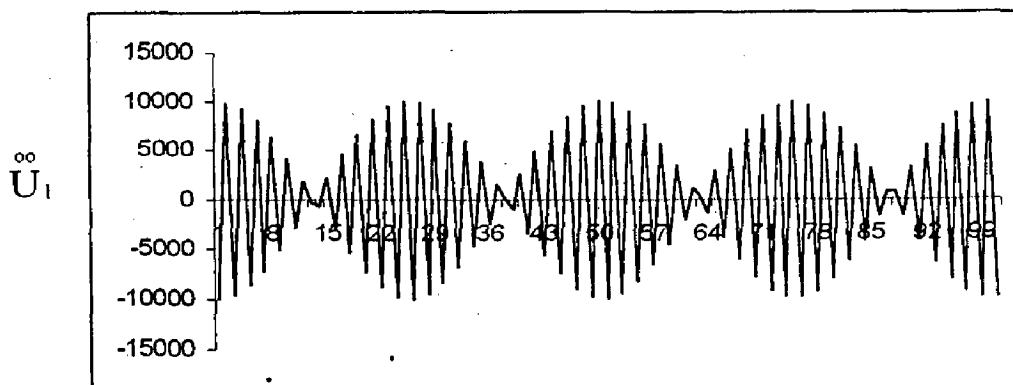


Figure 5.9 Acceleration vs. time steps

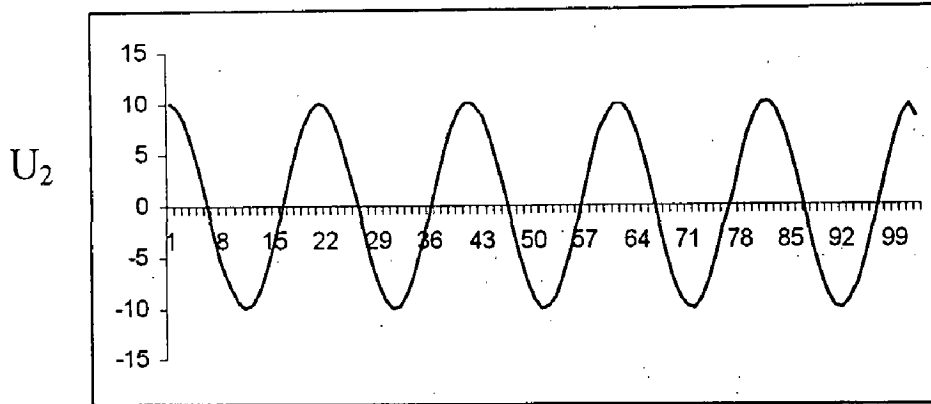


Figure 5.10 Displacement vs. time *steps*

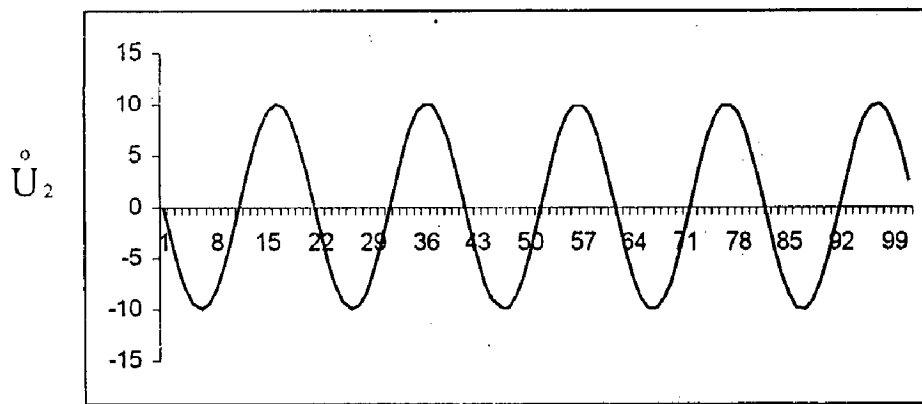


Figure 5.11 Velocity vs. time *steps*

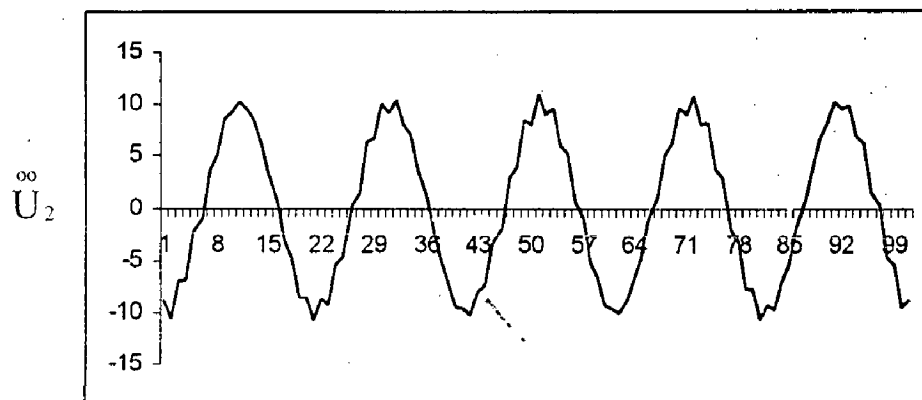


Figure 5.12 Acceleration vs. time ²*steps*

DAMPED NEWMARK METHOD

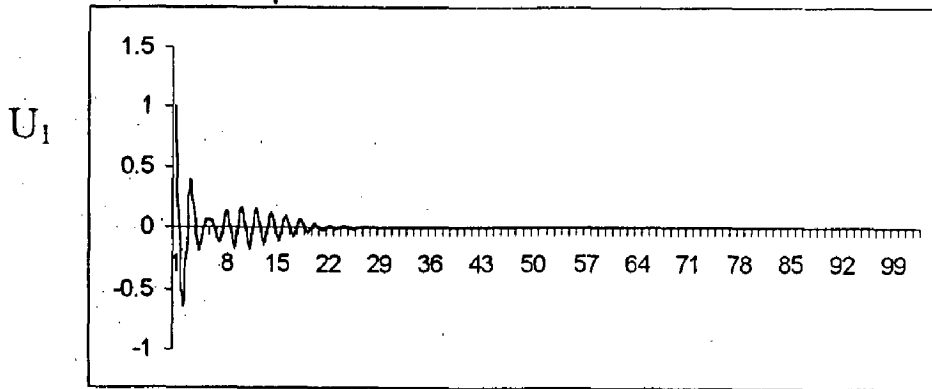


Figure 5.13 Displacement vs. time steps

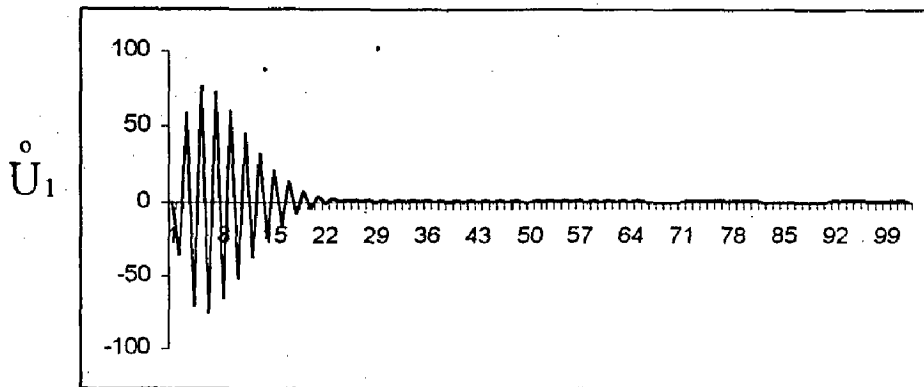


Figure 5.14 Velocity vs. time steps

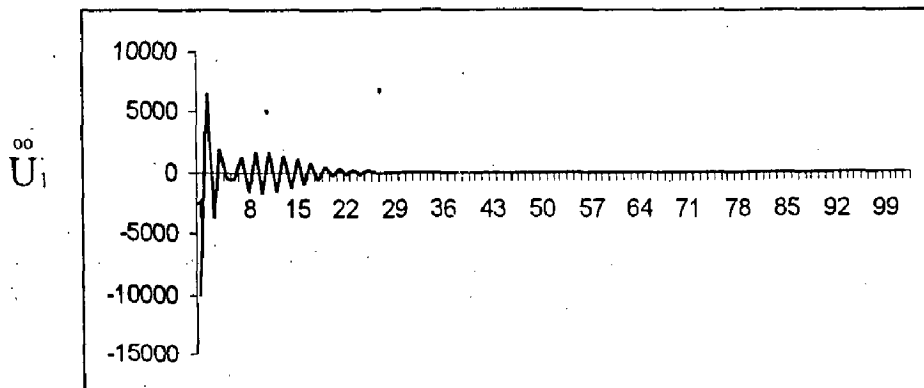


Figure 5.15 Acceleration vs. time steps

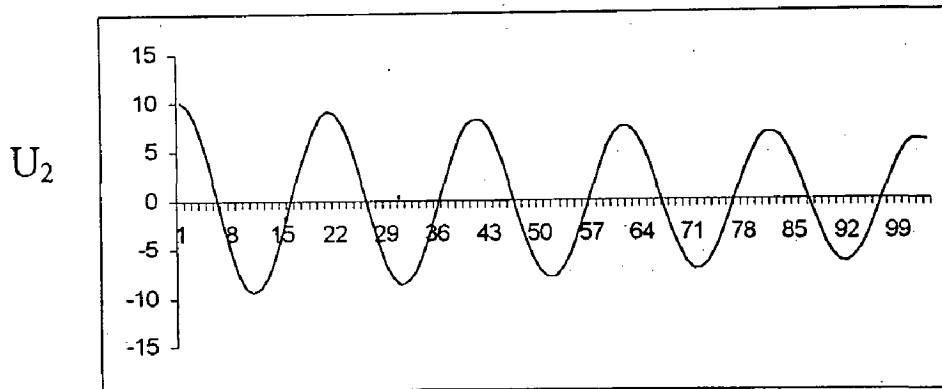


Figure 5.16 Displacement vs. time steps

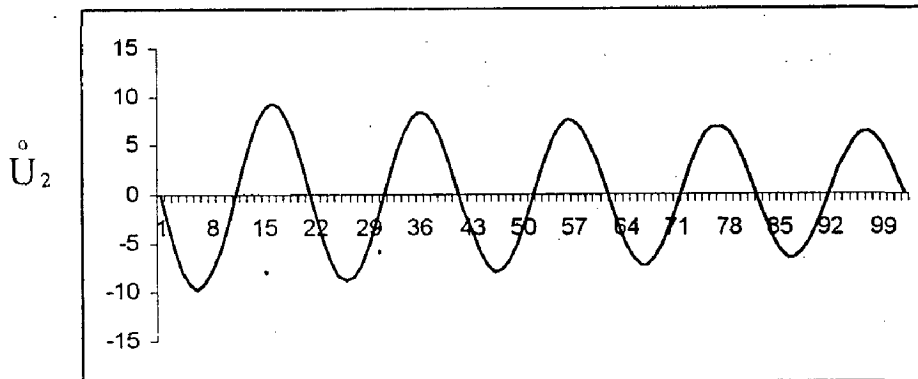


Figure 5.17 Velocity vs. time steps

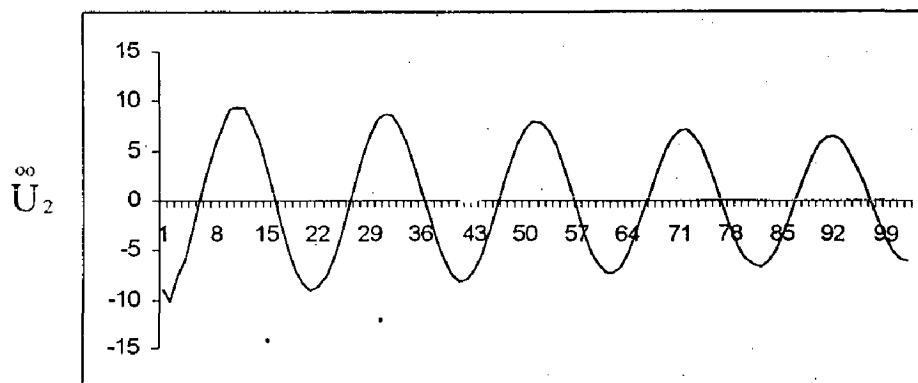


Figure 5.18 Acceleration vs. time steps

Houbolt Method

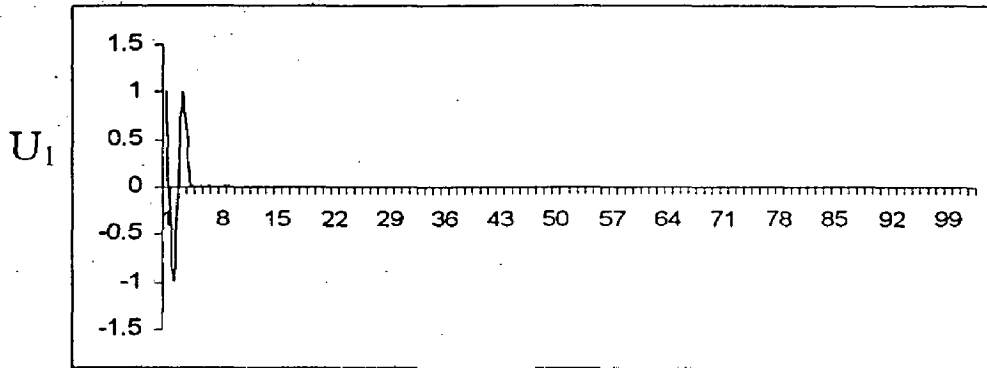


Figure 5.19 Displacement vs. time *steps*

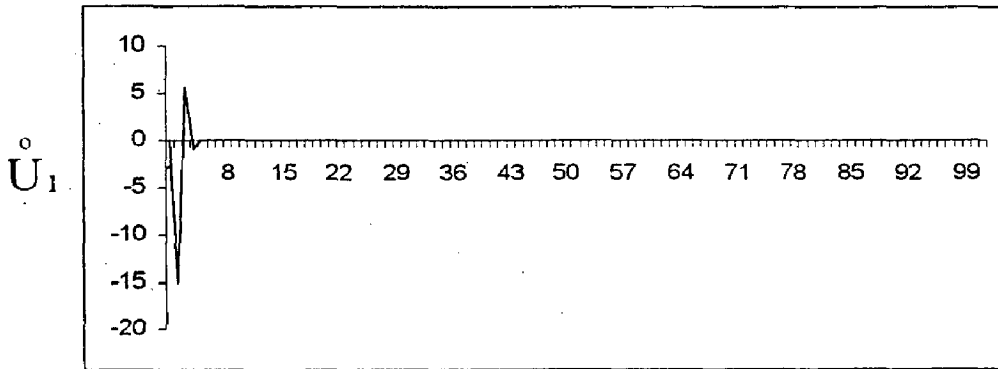


Figure 5.20 Velocity vs. time *steps*

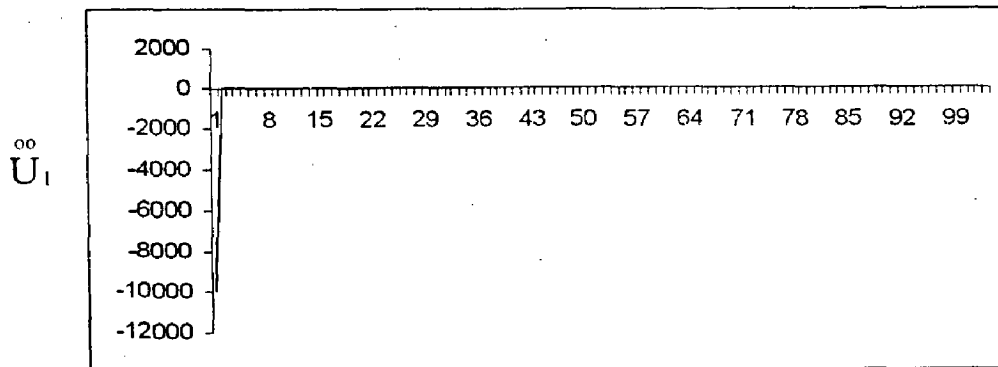


Figure 5.21 Acceleration vs. time *steps*

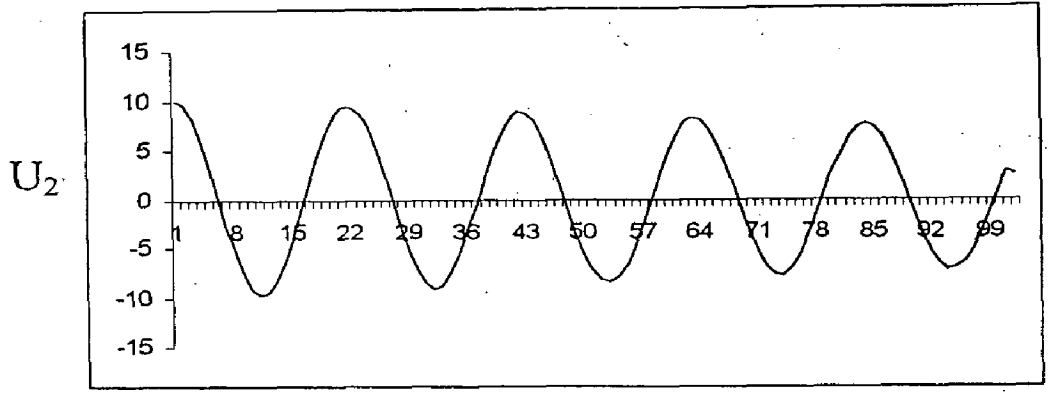


Figure 5.22 Displacement vs. time *steps*

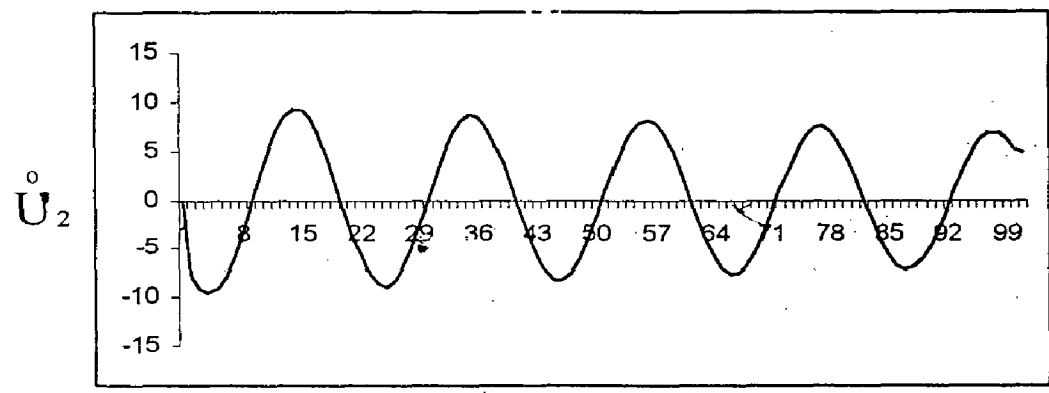


Figure 5.23 Velocity vs. time *steps*

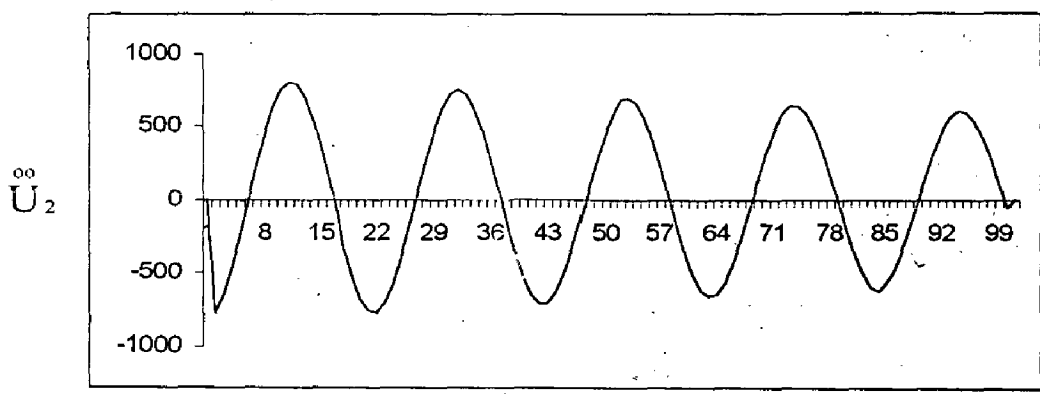


Figure 5.24 Acceleration vs. time *steps*

WILSON THETA METHOD

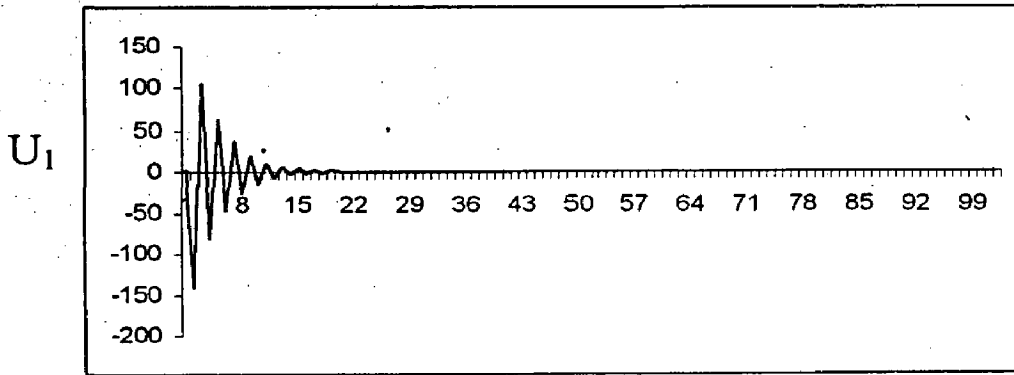


Figure 5.25 Displacement vs. time *steps*

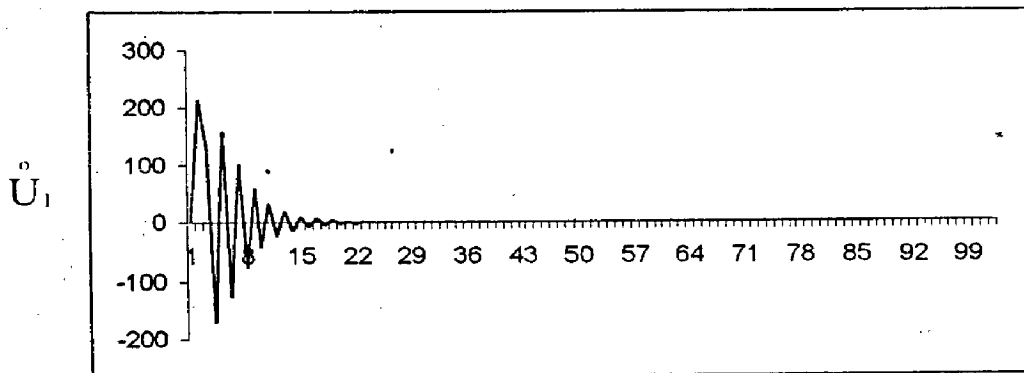


Figure 5.26 Velocity vs. time *steps*

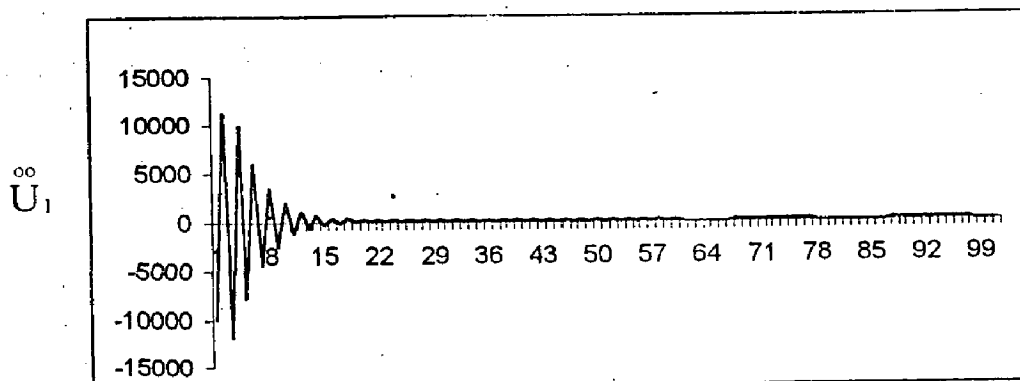


Figure 5.27 Acceleration vs. time *steps*

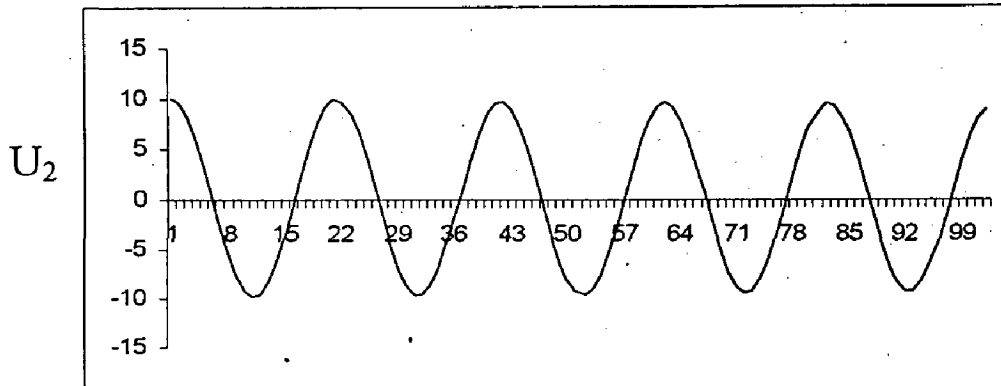


Figure 5.28 Displacement vs. time *steps*

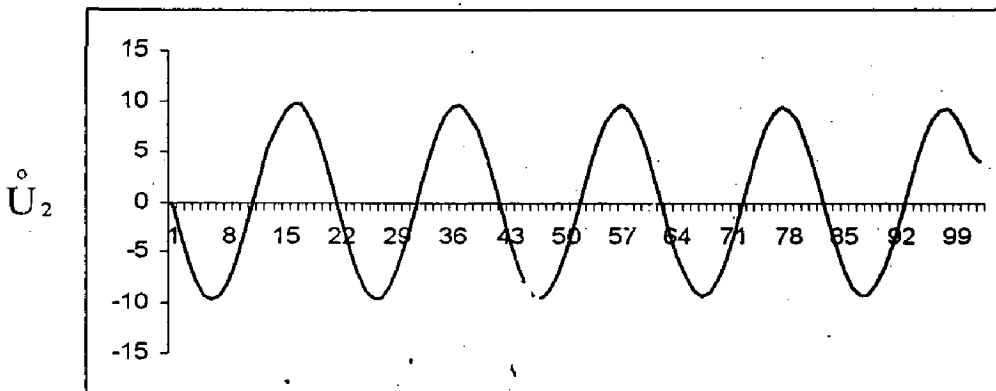


Figure 5.29 Velocity vs. time *steps*

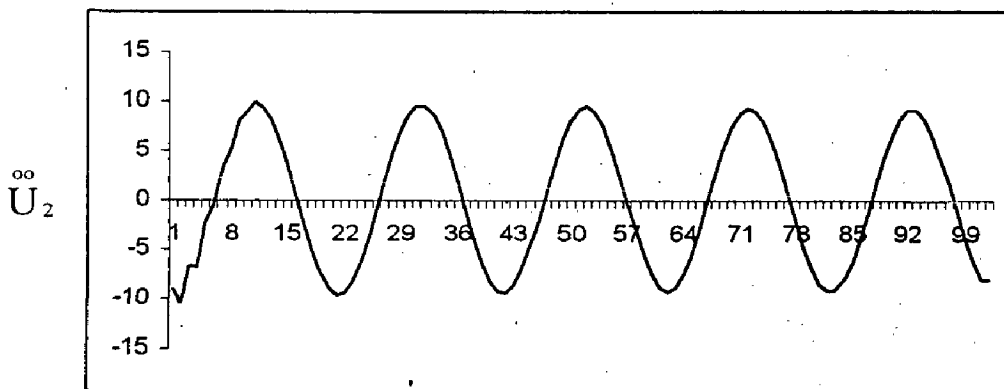


Figure 5.30 Acceleration vs. time *steps*

APPENDIX

*****CENTRAL DIFFERENCE METHOD*****

```
#include<fstream.h>
#include<iomanip.h>
#include"central.h"
#include<math.h>
#include<conio.h>

void main( )
{
clrscr();

int nr,nc;
float K[2][2],M[2][2],R[2][2],u0[2][2],u1[2][2],v0[2][2],acc0[2][2];
float u[100][2][2],v[100][2][2],acc[100][2][2];
float m[2][2],k[2][2],mi[2][2],MI[2][2];
float t=0.01999,a0=1/(t*t),a1=1/(2*t),a2=2*a0,a3=1/a2;

// ENTER NO.OF ROWS & COLS. FOR MASS MATRIX & STIFF. MATRIX
fin>>nr>>nc;

getmat(M,nr,nc);
fout<<"MASS MATRIX FOLLOWS----"<<endl;
printmat(M,nr,nc);
fout<<"NEW MASS MATRIX FOLLOWS----"<<endl;
nmassmat(M,nr,nc,a0,m);

getmat(K,nr,nc);
fout<<"STIFFNESS MATRIX FOLLOWS----"<<endl;
printmat(K,nr,nc);
fout<<"NEW STIFFNESS MATRIX FOLLOWS----"<<endl;
nstiffmat(nr,nc,a2,k,M,K);

// ENTER NO.OF ROWS & COLS. FOR LOADING MATRIX
fin>>nr>>nc;
getmat(R,nr,nc);
fout<<"LOADING MATRIX FOLLOWS----"<<endl;
printmat(R,nr,nc);

// ENTER NO. OF ROWS FOR DISPL. AND VELOCITY MATRIX
fin>>nr>>nc;

getmat(u0,nr,nc);
fout<<"INITIAL DISPLACEMENT MATRIX FOLLOWS----"<<endl;
printmat(u0,nr,nc);

getmat(v0,nr,nc);
fout<<"INITIAL VELOCITY MATRIX FOLLOWS----"<<endl;
printmat(v0,nr,nc);

fout<<"INITIAL ACCELERATION MATRIX FOLLOWS----"<<endl;
inverse(MI,M);
accmat(K,u0,acc0,MI,R);
```

```

fout<<"u(-t) MATRIX FOLLOWS----"<<endl;
    u1mat(u0,nr,nc,a3,acc0,u1);
fout<<"FINAL RESPONSE FOLLOW---"<<endl;
    inverse(mi,m);
    displmat(a0,a1,R,k,m,u,u0,v0,acc0,u1,mi,v,acc);

    fin.close();
    fout.close();
}

```

*******HEADER FILE*******

```

ifstream fin("central.dat");
ofstream fout("central.out");

// FUNCTION DEFINITION
void getmat(float a[2][2],int nr,int nc)
{
for(int i=0;i<nr;i++)
    {
        for(int j=0;j<nc;j++)
            {
                fin>>a[i][j];
            }
    }
}

void printmat(float a[2][2],int nr,int nc)
{
for(int i=0;i<nr;i++)
    {
        for(int j=0;j<nc;j++)
            {
                fout<<" "<<a[i][j];
            }
        fout<<endl;
    }
}

void nmassmat(float a[2][2],int nr,int nc,float a0,float m[2][2])
{
int i,j;
for(i=0;i<nr;i++)
    {
        for(j=0;j<nc;j++)
            {
                m[i][j]=a0*a[i][j];
                fout<<" "<<m[i][j];
            }
        fout<<endl;
    }
}

```

```

    }
}

void nstiffmat(int nr,int nc, float a2,float k[2][2],
float M[2][2],float K[2][2])
{
for(int i=0;i<nr;i++)
{
for(int j=0;j<nc;j++)
{
k[i][j]=K[i][j]-a2*M[i][j];
fout<<" "<<k[i][j];
}
fout<<endl;
}
}

```

```

void accmat(float K[2][2],float u0[2][2],float acc0[2][2],
float MI[2][2],float R[2][2])
{
float d[5][5];
for(int i=0;i<2;i++)
{
for(int j=0;j<1;j++)
{
d[i][j]=0;
for(int l=0;l<2;l++)
{
d[i][j]=d[i][j]+K[i][l]*u0[l][j];
}}}
}
}

```

```

for(int i=0;i<2;i++)
{
for(int j=0;j<1;j++)
{
acc0[i][j]=0;
for(int l=0;l<2;l++)
{
acc0[i][j]=acc0[i][j]+MI[i][l]*(-d[l][j])+MI[i][l]*R[l][j];
}
fout<<" "<<acc0[i][j];
}
}
fout<<endl;
}
}

```

```

void ulmat(float u0[2][2],int nr,int nc,float a3,
float acc0[2][2],float u1[2][2])
{
for(int i=0;i<nr;i++)
{
for(int j=0;j<nc;j++)
{
u1[i][j]=u0[i][j]+a3*acc0[i][j];
}
}
}

```

```

        fout<<" "<<u1[i][j];
    }
    fout<<endl;
}
}

```

```

void inverse(float b1[2][2],float a1[2][2])

```

```

{
    float c;
    c=1.0/(a1[0][0]*a1[1][1]-a1[0][1]*a1[1][0]);
    b1[0][0]=a1[1][1]*c;
    b1[0][1]=-a1[0][1]*c;
    b1[1][0]=-a1[1][0]*c;
    b1[1][1]=a1[0][0]*c;

    for(int i=0;i<2;i++)
    {
        for(int j=0;j<2;j++)
        {
            b1[i][j];
        }
    }
}

```

```

void displmat(float a0,float a1,float R[2][2],float k[2][2],
float m[2][2],float u[100][2][2],float u0[2][2],float v0[2][2],
float acc0[2][2],float u1[2][2],float b1[2][2],
float v[100][2][2],float acc[100][2][2] )

```

```

{
    float A[100][2][2],B[100][2][2],R1[100][2][2];
    for(int p=1;p<=100;p++)
    {
        for(int i=0;i<2;i++)
        {
            for(int j=0;j<1;j++)
            {
                A[p][i][j]=0.0;
                B[p-1][i][j]=0.0;
                u[p+1][i][j]=0;
                for(int l=0;l<2;l++)
                {
                    u[l][i][j]=u0[l][j];
                    u[0][i][j]=u1[l][j];
                    A[p][i][j]=A[p][i][j]+k[i][l]*u[p][l][j];
                    B[p-1][i][j]=B[p-1][i][j]+m[i][l]*u[p-1][l][j];
                    R1[p][i][j]=R[i][j]-A[p][i][j]-B[p-1][i][j];
                }
            }
        }
        for(int i=0;i<2;i++)
        {
            for(int j=0;j<1;j++)
            {
                for(int l=0;l<2;l++)
                {

```



```

        u[p+1][i][j]=u[p+1][i][j]+b1[i][1]*R1[p][1][j];
    }
    v[p][i][j]= a1*(-u[p-1][i][j]+u[p+1][i][j]);
    acc[p][i][j]=a0*(u[p-1][i][j]-2*u[p][i][j]
        +u[p+1][i][j]);
    }}}
    for(int p=0;p<=100;p++)
    {
        for(int i=0;i<2;i++)
        {
            for(int j=0;j<1;j++)
            {
                fout<<setiosflags(ios::fixed)<<setiosflags(ios::showpoint)
                    <<setprecision(4)<<setw(20)<<u[p+1][i][j];

            }

        }
        fout<<endl;
    }
    fout<<endl<<endl;
    for(int p=0;p<=100;p++)
    {
        for(int i=0;i<2;i++)
        {
            for(int j=0;j<1;j++)
            {
                v[1][i][j]=v0[i][j];
                fout<<setiosflags(ios::fixed)<<setiosflags(ios::showpoint)
                    <<setprecision(4)<<setw(20)<<v[p+1][i][j];
            }

        }
        fout<<endl;
    }
    fout<<endl<<endl;
    for(int p=0;p<=100;p++)
    {
        for(int i=0;i<2;i++)
        {
            for(int j=0;j<1;j++)
            {
                acc[1][i][j]=acc0[i][j];
                fout<<setiosflags(ios::fixed)<<setiosflags(ios::showpoint)
                    <<setprecision(4)<<setw(20)<<acc[p+1][i][j];
            }

        }
        fout<<endl;
    }
}

```



*****INPUT DATA FILE*****

2
2
1
0
0
1
10001
-1
-1
1
2
1
0
0
2
1
1
10
0
0

THE NEWMARK METHOD (TRAPEZOIDAL RULE)

```
#include<fstream.h>
#include<iomanip.h>
#include"newmark.h"
#include<math.h>
#include<conio.h>

void main( )
{
clrscr( );

int nr,nc;
float K[2][2],M[2][2],R[2][2],u0[2][2],v0[2][2],acc0[2][2];
float u[100][2][2],v[100][2][2],acc[100][2][2];
float m1[2][2],m2[2][2],m3[2][2],k[2][2],ki[2][2],MI[2][2];
float t=0.314159,x=0.25,y=0.50,a0=1/(x*t*t),a2=1/(x*t),

/* Take x=0.3025; y=0.6 [ FOR DAMPED NEWMARK METHOD]*/

a3=(1/(2*x))-1,a6=t*(1-y),a7=y*t;
float a1=y/(x*t),a4=y/x-1,a5=(t/2)*(y/x-2);

//ENTER NO.OF ROWS & COLS. FOR MASS MATRIX & STIFF. MATRIX
fin>>nr>>nc;

getmat(M,nr,nc);
fout<<"MASS MATRIX FOLLOWS----"<<endl;
printmat(M,nr,nc);
fout<<"FIRST NEW MASS MATRIX FOLLOWS----"<<endl;
nmass1mat(M,nr,nc,a0,m1);
fout<<"SECOND NEW MASS MATRIX FOLLOWS----"<<endl;
nmass2mat(M,nr,nc,a2,m2);
fout<<"THIRD NEW MASS MATRIX FOLLOWS----"<<endl;
nmass3mat(M,nr,nc,a3,m3);

getmat(K,nr,nc);
fout<<"STIFFNESS MATRIX FOLLOWS----"<<endl;
printmat(K,nr,nc);
fout<<"NEW STIFFNESS MATRIX FOLLOWS----"<<endl;
nstiffmat(nr,nc,a0,k,M,K);

//ENTER NO.OF ROWS & COLS. FOR LOADING MATRIX
fin>>nr>>nc;
getmat(R,nr,nc);
fout<<"LOADING MATRIX FOLLOWS----"<<endl;
printmat(R,nr,nc);

//ENTER NO. OF ROWS FOR DISPL. AND VELOCITY MATRIX
fin>>nr>>nc;
getmat(u0,nr,nc);
fout<<"INITIAL DISPLACEMENT MATRIX FOLLOWS----"<<endl;
printmat(u0,nr,nc);
```

```

        getmat(v0,nr,nc);
fout<<"INITIAL VELOCITY MATRIX FOLLOWS----"<<endl;
        printmat(v0,nr,nc);

fout<<"INITIAL ACCELERATION MATRIX FOLLOWS----"<<endl;
        inverse(MI,M);
        accmat(K,u0,acc0,MI,R);

fout<<"FINAL RESPONSE FOLLOW---"<<endl;
        inverse(ki,k);
        displmat(a0,a2,a3,a6,a7,R,m1,m2,m3,u,u0,v,v0,acc,acc0,ki);

fin.close();
fout.close();

}

```

*******HEADER FILE*******

```

ifstream fin("newmark.dat");
ofstream fout("newmark.out");

//FUNCTION DEFINITION
void getmat(float a[2][2],int nr,int nc)
{
for(int i=0;i<nr;i++)
    {
        for(int j=0;j<nc;j++)
            {
                fin>>a[i][j];
            }
    }
}

void printmat(float a[2][2],int nr,int nc)
{
int i,j;
for(i=0;i<nr;i++)
    {
        for(j=0;j<nc;j++)
            {
                fout<<" "<<a[i][j];
            }
        fout<<endl;
    }
}

void nmass1mat(float M[2][2],int nr,int nc,float a0,float m1[2][2])
{
for(int i=0;i<nr;i++)

```

```

        {
            for(int j=0;j<nc;j++)
            {
                m1[i][j]=a0*M[i][j];
                fout<<" "<<m1[i][j];
            }
            fout<<endl;
        }
    }

void nmass2mat(float M[2][2],int nr,int nc,float a2,float m2[2][2])
{
    for(int i=0;i<nr;i++)
    {
        for(int j=0;j<nc;j++)
        {
            m2[i][j]=a2*M[i][j];
            fout<<" "<<m2[i][j];
        }
        fout<<endl;
    }
}

void nmass3mat(float M[2][2],int nr,int nc,float a3,float m3[2][2])
{
    for(int i=0;i<nr;i++)
    {
        for(int j=0;j<nc;j++)
        {
            m3[i][j]=a3*M[i][j];
            fout<<" "<<m3[i][j];
        }
        fout<<endl;
    }
}

void nstiffmat(int nr,int nc, float a0,float k[2][2],float M[2][2],
float K[2][2])
{
    for(int i=0;i<nr;i++)
    {
        for(int j=0;j<nc;j++)
        {
            k[i][j]=K[i][j]+a0*M[i][j];
            fout<<" "<<k[i][j];
        }
        fout<<endl;
    }
}

void accmat(float K[2][2],float u0[2][2],float acc0[2][2],
float MI[2][2],float R[2][2])
{

```

```

float d[2][2];
for(int i=0;i<2;i++)
    {
        for(int j=0;j<1;j++)
            {
                d[i][j]=0;
                for(int l=0;l<2;l++)
                    {
                        d[i][j]=d[i][j]+K[i][l]*u0[l][j];
                    }
            }
    }

for(int i=0;i<2;i++)
    {
        for(int j=0;j<1;j++)
            {
                acc0[i][j]=0;
                for(int l=0;l<2;l++)
                    {
                        acc0[i][j]=acc0[i][j]+MI[i][l]*(-d[l][j])+MI[i][l]*R[l][j];
                    }
                fout<<" "<<acc0[i][j];
            }
        fout<<endl;
    }
}

```

```

void inverse(float b1[2][2],float a1[2][2])
{
    float c;
    c=1.0/(a1[0][0]*a1[1][1]-a1[0][1]*a1[1][0]);
    b1[0][0]=a1[1][1]*c;
    b1[0][1]=-a1[0][1]*c;
    b1[1][0]=-a1[1][0]*c;
    b1[1][1]=a1[0][0]*c;

    for(int i=0;i<2;i++)
        {
            for(int j=0;j<2;j++)
                {
                    b1[i][j];
                }
        }
}

```

```

void displmat(float a0,float a2,float a3,float a6,float a7,
float R[2][2],float m1[2][2],float m2[2][2],float m3[2][2],
float u[100][2][2],float u0[2][2],float v[100][2][2],float v0[2][2],
float acc[100][2][2],float acc0[2][2],float b1[2][2])
{
    float A[100][2][2],B[100][2][2],C[100][2][2],R1[100][2][2];
    for(int p=1;p<=100;p++)
        {
            for(int i=0;i<2;i++)

```

```

{
for(int j=0;j<1;j++)
{
A[p][i][j]=0.0;
B[p][i][j]=0.0;
C[p][i][j]=0.0;
u[p+1][i][j]=0.0;
for(int l=0;l<2;l++)
{
u[l][i][j]=u0[l][j];
v[l][i][j]=v0[l][j];
acc[l][i][j]=acc0[l][j];
A[p][i][j]=A[p][i][j]+m1[i][l]*u[p][l][j];
B[p][i][j]=B[p][i][j]+m2[i][l]*v[p][l][j];
C[p][i][j]=C[p][i][j]+m3[i][l]*acc[p][l][j];
R1[p][i][j]=R[i][j]+A[p][i][j]+B[p][i][j]+C[p][i][j];
}}}
for(int i=0;i<2;i++)
{
for(int j=0;j<1;j++)
{
for(int l=0;l<2;l++)
{
u[p+1][i][j]=u[p+1][i][j]+b1[i][l]*R1[p][l][j];
acc[p+1][i][j]=a0*(u[p+1][i][j]-u[p][i][j])
-a2*v[p][i][j]-a3*acc[p][i][j];
v[p+1][i][j]=v[p][i][j]+a6*acc[p][i][j]
+a7*acc[p+1][i][j];
}}}}
for(int p=0;p<=100;p++)
{
for(int i=0;i<2;i++)
{
for(int j=0;j<1;j++)
{
fout<<setiosflags(ios::fixed)
<<setiosflags(ios::showpoint)<<setprecision(4)<<setw(20)
<<u[p+1][i][j];
}
}
fout<<endl;
}
fout<<endl<<endl;
for(int p=0;p<=100;p++)
{
for(int i=0;i<2;i++)
{
for(int j=0;j<1;j++)
{
fout<<setiosflags(ios::fixed)
<<setiosflags(ios::showpoint)<<setprecision(4)<<setw(20)
<<v[p+1][i][j];
}
}
}
}

```

```

        fout<<endl;
    }
    fout<<endl<<endl;
    for(int p=0;p<=100;p++)
    {
        for(int i=0;i<2;i++)
        {
            for(int j=0;j<1;j++)
            {
                fout<<setiosflags(ios::fixed)
                    <<setiosflags(ios::showpoint)<<setprecision(4)<<setw(20)
                    <<acc[p+1][i][j];
            }
        }
        fout<<endl;
    }
}

```

*****INPUT DATA FILE*****

```

2
2
1
0
0
1
10001
-1
-1
1
2
1
0
0
2
1
1
10
0
0

```


***** HOUBOLT METHOD *****

```

#include<fstream.h>
#include<iomanip.h>
#include"houbolt.h"
#include<math.h>
#include<conio.h>

void main( )
{
clrscr( );

int nr,nc;
float K[2][2],M[2][2],MI[2][2],u0[2][2],R[2][2],v0[2][2],acc0[2][2];
float u[100][2][2],v[100][2][2],acc[100][2][2];
float m1[2][2],m2[2][2],m3[2][2],k[2][2],u1[2][2],u2[2][2],ki[2][2];
float t=0.314159,a0=2/(t*t),a1=11/(6*t),a2=5/(t*t),a3=3/t,a4=2*a0,
a5=-0.5*a3,a6=0.5*a0,a7=a3/9.0;

//ENTER NO. OF ROWS & COLS. FOR MASS & STIFFNESS MATRIX
fin>>nr>>nc;

getmat(M,nr,nc);
fout<<"MASS MATRIX FOLLOWS----"<<endl;
printmat(M,nr,nc);
fout<<"FIRST NEW MASS MATRIX FOLLOWS----"<<endl;
nmass1mat(M,nr,nc,a2,m1);
fout<<"SECOND NEW MASS MATRIX FOLLOWS----"<<endl;
nmass2mat(M,nr,nc,a4,m2);
fout<<"THIRD NEW MASS MATRIX FOLLOWS----"<<endl;
nmass3mat(M,nr,nc,a6,m3);

getmat(K,nr,nc);
fout<<"STIFFNESS MATRIX FOLLOWS----"<<endl;
printmat(K,nr,nc);
fout<<"NEW STIFFNESS MATRIX FOLLOWS----"<<endl;
nstiffmat(nr,nc,a0,k,M,K);

//ENTER NO. OF ROWS & COLS. FOR LOADING MATRIX
fin>>nr>>nc;
getmat(R,nr,nc);
fout<<"LOADING MATRIX FOLLOWS----"<<endl;
printmat(R,nr,nc);

//ENTER NO.OF ROWS & COLS. FOR DISPL.& VELOCITY MATRIX
fin>>nr>>nc;
getmat(u0,nr,nc);
fout<<"DISPLACEMENT MATRIX UO FOLLOWS----"<<endl;
printmat(u0,nr,nc);

getmat(v0,nr,nc);
fout<<"VELOCITY MATRIX VO FOLLOWS----"<<endl;
printmat(v0,nr,nc);

```

```

fout<<"INITIAL ACCELERATION MATRIX FOLLOWS----"<<endl;
    inverse(MI,M);
    accmat(K,u0,acc0,MI,R);

//FIRST TWO DISPL. MATRIX AS OBTAINED BY TRAPEZOIDAL METHOD
    getmat(u1,nr,nc);
fout<<"DISPLACEMENT MATRIX U1 FOLLOWS----"<<endl;
    printmat(u1,nr,nc);

    getmat(u2,nr,nc);
fout<<"DISPLACEMENT MATRIX U2 FOLLOWS----"<<endl;
    printmat(u2,nr,nc);

fout<<"FINAL RESPONSE FOLLOW---"<<endl;
    inverse(ki,k);
    displmat(a0,a1,a2,a3,a4,a5,a6,a7,R,m1,m2,m3,u,u0,u1,u2,v0,acc0,v,acc,ki);

    fin.close();
    fout.close();
}

```

*******HEADER FILE*******

```

ifstream fin("houbolt.dat");
ofstream fout("houbolt.out");

//FUNCTION DEFINITION
void getmat(float a[2][2],int nr,int nc)
{
for(int i=0;i<nr;i++)
    {
        for(int j=0;j<nc;j++)
            {
                fin>>a[i][j];
            }
    }
}

void printmat(float a[2][2],int nr,int nc)
{
int i,j;
for(i=0;i<nr;i++)
    {
        for(j=0;j<nc;j++)
            {
                fout<<" "<<a[i][j];
            }
        fout<<endl;
    }
}

```

```

void nmass1mat(float a[2][2],int nr,int nc,float a2,float m1[2][2])
{
for(int i=0;i<nr;i++)
{
for(int j=0;j<nc;j++)
{
m1[i][j]=a2*a[i][j];
fout<<" "<<m1[i][j];
}
fout<<endl;
}
}

```

```

void nmass2mat(float a[2][2],int nr,int nc,float a4,float m2[2][2])
{
for(int i=0;i<nr;i++)
{
for(int j=0;j<nc;j++)
{
m2[i][j]=a4*a[i][j];
fout<<" "<<m2[i][j];
}
fout<<endl;
}
}

```

```

void nmass3mat(float a[2][2],int nr,int nc,float a6,float m3[2][2])
{
for(int i=0;i<nr;i++)
{
for(int j=0;j<nc;j++)
{
m3[i][j]=a6*a[i][j];
fout<<" "<<m3[i][j];
}
fout<<endl;
}
}

```

```

void nstiffmat(int nr,int nc, float a0,float k[2][2],
float M[2][2],float K[2][2])
{
for(int i=0;i<nr;i++)
{
for(int j=0;j<nc;j++)
{
k[i][j]=K[i][j]+a0*M[i][j];
fout<<" "<<k[i][j];
}
fout<<endl;
}
}

```

```

    }
}

void accmat(float K[2][2],float u0[2][2],float acc0[2][2],
float MI[2][2],float R[2][2])
{
float d[5][5];
for(int i=0;i<2;i++)
    {
        for(int j=0;j<1;j++)
            {
                d[i][j]=0;
                for(int l=0;l<2;l++)
                    {
                        d[i][j]=d[i][j]+K[i][l]*u0[l][j];
                    }
            }
    }

for(int i=0;i<2;i++)
    {
        for(int j=0;j<1;j++)
            {
                acc0[i][j]=0;
                for(int l=0;l<2;l++)
                    {
                        acc0[i][j]=acc0[i][j]+MI[i][l]*(-d[l][j])+MI[i][l]*R[l][j];
                    }
                . fout<<" " <<acc0[i][j];
            }
        fout<<endl;
    }
}

void inverse(float b1[2][2],float a1[2][2])
{
float c;
c=1.0/(a1[0][0]*a1[1][1]-a1[0][1]*a1[1][0]);
b1[0][0]=a1[1][1]*c;
b1[0][1]=-a1[0][1]*c;
b1[1][0]=-a1[1][0]*c;
b1[1][1]=a1[0][0]*c;

for(int i=0;i<2;i++)
    {
        for(int j=0;j<2;j++)
            {
                b1[i][j];
            }
    }
}

void displmat(float a0,float a1,float a2,float a3,float a4,
float a5,float a6,float a7,float R[2][2],float m1[2][2],
float m2[2][2],float m3[2][2],float u[100][2][2],float u0[2][2],
float u1[2][2],float u2[2][2],float v0[2][2],float acc0[2][2],
float v[100][2][2],float acc[100][2][2],float b1[2][2])
{

```

```

float A[100][2][2],B[100][2][2],C[100][2][2],R1[100][2][2];
for(int p=2;p<=100;p++)
{
    for(int i=0;i<2;i++)
    {
        for(int j=0;j<1;j++)
        {
            A[p][i][j]=0.0;
            B[p-1][i][j]=0.0;
            C[p-2][i][j]=0.0;
            for(int l=0;l<2;l++)
            {
                u[2][l][j]=u2[l][j];
                u[1][l][j]=u1[l][j];
                u[0][l][j]=u0[l][j];

                A[p][i][j]=A[p][i][j]+m1[i][l]*u[p][l][j];
                B[p-1][i][j]=B[p-1][i][j]+m2[i][l]*u[p-1][l][j];
                C[p-2][i][j]=C[p-2][i][j]+m3[i][l]*u[p-2][l][j];
                R1[p][i][j]=R[i][j]+A[p][i][j]-B[p-1][i][j]+C[p-2][i][j];
            }}
            for(int i=0;i<2;i++)
            {
                for(int j=0;j<1;j++)
                {
                    u[p+1][i][j]=0.0;
                    for(int l=0;l<2;l++)
                    {
                        u[p+1][i][j]=u[p+1][i][j]+b1[i][l]*R1[p][l][j];
                    }
                    v[p+1][i][j]=a1*u[p+1][i][j]-a3*u[p][i][j]
                        -a5*u[p-1][i][j]-a7*u[p-2][i][j];
                    acc[p+1][i][j]=a0*u[p+1][i][j]-a2*u[p][i][j]
                        -a4*u[p-1][i][j]-a6*u[p-2][i][j];
                }}
            for(int p=0;p<=100;p++)
            {
                for(int i=0;i<2;i++)
                {
                    for(int j=0;j<1;j++)
                    {
                        fout<<setiosflags(ios::fixed)
                            <<setiosflags(ios::showpoint)<<setprecision(4)
                            <<setw(20)<<u[p][i][j];
                    }
                }
                fout<<endl;
            }
            fout<<endl<<endl;

            for(int p=0;p<=100;p++)
            {
                for(int i=0;i<2;i++)
                {
                    for(int j=0;j<1;j++)
                    {

```

```

        v[2][i][j]=v0[i][j];
        fout<<setiosflags(ios::fixed)
            <<setiosflags(ios::showpoint)
            <<setprecision(4)<<setw(20)<<v[p+2][i][j];
    }

    }
    fout<<endl;
}
fout<<endl<<endl;
for(int p=0;p<100;p++)
{
    for(int i=0;i<2;i++)
    {
        for(int j=0;j<1;j++)
        {
            acc[2][i][j]=acc0[i][j];
            fout<<setiosflags(ios::fixed)
                <<setiosflags(ios::showpoint)
                <<setprecision(4)<<setw(20)<<acc[p+2][i][j];
        }
    }
    fout<<endl;
}
}

```

*****INPUT DATA FILE*****

```

2
2
1
0
0
1
10001
-1
-1
1
2
1
0
0
2
1
1
10
0
0
-0.99
9.5186
0.9677
8.1202

```

*****WILSON THETA METHOD*****

```

#include<fstream.h>
#include<iomanip.h>
#include"Wilson.h"
#include<math.h>
#include<conio.h>

void main()
{
    clrscr( );

    int nr,nc;
    float K[2][2],M[2][2],R[2][2],u0[2][2],v0[2][2],acc0[2][2];
    float u[100][2][2],v[100][2][2],acc[100][2][2];
    float m1[2][2],m2[2][2],m3[2][2],k[2][2],ki[2][2],MI[2][2];
    float t=0.314159,h=1.40,a0=6/(h*h*t*t),a1=3/(h*t),a2=2*a1;
    float a4=a0/h,a5=-a2/h,a6=1-(3/h),a7=0.5*t,a8=(t*t)/6;
    //float a3=0.5*h*t;

    //ENTER NO. OF ROWS & COLS. FOR MASS & STIFFNESS MATRIX
    fin>>nr>>nc;

    getmat(M,nr,nc);
    fout<<"MASS MATRIX FOLLOWS----"<<endl;
    printmat(M,nr,nc);
    fout<<"FIRST NEW MASS MATRIX FOLLOWS----"<<endl;
    nmass1mat(M,nr,nc,a0,m1);
    fout<<"SECOND NEW MASS MATRIX FOLLOWS----"<<endl;
    nmass2mat(M,nr,nc,a2,m2);
    fout<<"THIRD NEW MASS MATRIX FOLLOWS----"<<endl;
    nmass3mat(M,nr,nc,m3);

    getmat(K,nr,nc);
    fout<<"STIFFNESS MATRIX FOLLOWS----"<<endl;
    printmat(K,nr,nc);
    fout<<"NEW STIFFNESS MATRIX FOLLOWS----"<<endl;
    nstiffmat(nr,nc,a0,k,M,K);

    //ENTER NO. OF ROWS & COLS. FOR LOADING MATRIX
    fin>>nr>>nc;
    getmat(R,nr,nc);
    fout<<"LOADING MATRIX FOLLOWS----"<<endl;
    printmat(R,nr,nc);

    //ENTER NO. OF ROWS & COLS. FOR DISPL. AND VELOCITY MATRIX
    fin>>nr>>nc;

    getmat(u0,nr,nc);
    fout<<"DISPLACEMENT MATRIX UO FOLLOWS----"<<endl;
    printmat(u0,nr,nc);

```

```

        getmat(v0,nr,nc);
fout<<"VELOCITY MATRIX V0 FOLLOWS----"<<endl;
        printmat(v0,nr,nc);

fout<<"INITIAL ACCELERATION MATRIX FOLLOWS----"<<endl;
        inverse(MI,M);
        accmat(K,u0,acc0,MI,R);

fout<<"FINAL RESPONSE FOLLOW---"<<endl;
        inverse(ki,k);
        displmat(a4,a5,a6,a7,a8,t,R,m1,m2,m3,u,u0,v,v0,acc,acc0,ki);

fin.close();
fout.close();

}

```

*******HEADER FILE*******

```

ifstream fin("wilson.dat");
ofstream fout("wilson.out");

//FUNCTION DEFINITION
void getmat(float a[2][2],int nr,int nc)
{
for(int i=0;i<nr;i++)
    {
        for(int j=0;j<nc;j++)
            {
                fin>>a[i][j];
            }
    }
}

void printmat(float a[2][2],int nr,int nc)
{
for(int i=0;i<nr;i++)
    {
        for(int j=0;j<nc;j++)
            {
                fout<<" "<<a[i][j];
            }
        fout<<endl;
    }
}

void nmasslmat(float M[2][2],int nr,int nc,float a0,float m1[2][2])
{
for(int i=0;i<nr;i++)
    {
        for(int j=0;j<nc;j++)
            {

```



```

        m1[i][j]=a0*M[i][j];
        fout<<" "<<m1[i][j];
    }
    fout<<endl;
}

void nmass2mat(float M[2][2],int nr,int nc,float a2,float m2[2][2])
{
    for(int i=0;i<nr;i++)
    {
        for(int j=0;j<nc;j++)
        {
            m2[i][j]=a2*M[i][j];
            fout<<" "<<m2[i][j];
        }
        fout<<endl;
    }
}

void nmass3mat(float M[2][2],int nr,int nc,float m3[2][2])
{
    for(int i=0;i<nr;i++)
    {
        for(int j=0;j<nc;j++)
        {
            m3[i][j]=2.0*M[i][j];
            fout<<" "<<m3[i][j];
        }
        fout<<endl;
    }
}

void nstiffmat(int nr,int nc, float a0,float k[2][2],
float M[2][2],float K[2][2])
{
    for(int i=0;i<nr;i++)
    {
        for(int j=0;j<nc;j++)
        {
            k[i][j]=K[i][j]+a0*M[i][j];
            fout<<" "<<k[i][j];
        }
        fout<<endl;
    }
}

void accmat(float K[2][2],float u0[2][2],float acc0[2][2],
float MI[2][2],float R[2][2])
{
    float d[2][2];
    for(int i=0;i<2;i++)
    {

```

```

        for(int j=0;j<1;j++)
        {
d[i][j]=0;
        for(int l=0;l<2;l++)
            {
                d[i][j]=d[i][j]+K[i][l]*u0[l][j];
            }
    }

for(int i=0;i<2;i++)
    {
        for(int j=0;j<1;j++)
            {
                acc0[i][j]=0;
                for(int l=0;l<2;l++)
                    {
                        acc0[i][j]=acc0[i][j]+MI[i][l]*-d[l][j]+MI[i][l]*R[l][j];
                    }
                fout<<" "<<acc0[i][j];
            }
        fout<<endl;
    }
}

```

```

void inverse(float b1[2][2],float a1[2][2])
{
    float c;
    c=1.0/(a1[0][0]*a1[1][1]-a1[0][1]*a1[1][0]);
    b1[0][0]=a1[1][1]*c;
    b1[0][1]=-a1[0][1]*c;
    b1[1][0]=-a1[1][0]*c;
    b1[1][1]=a1[0][0]*c;

    for(int i=0;i<2;i++)
    {
        for(int j=0;j<2;j++)
        {
            b1[i][j];
        }
    }
}

```

```

void displmat(float a4,float a5,float a6,float a7,float a8,float t,
float R[2][2],float m1[2][2],float m2[2][2],float m3[2][2],
float u[100][2][2],float u0[2][2],float v[100][2][2],float v0[2][2],
float acc[100][2][2],float acc0[2][2],float b1[2][2])
{
    float A[100][2][2],B[100][2][2],C[100][2][2],R1[100][2][2];
    for(int p=1;p<=100;p++)
    {
        for(int q=1;q<=101;q++)
        {
            for(int p=1;p<=100;p++)
            {
                for(int i=0;i<2;i++)
                {
                    for(int j=0;j<1;j++)

```

```

{
A[p][i][j]=0.0;
B[p][i][j]=0.0;
C[p][i][j]=0.0;
for(int l=0;l<2;l++)
{
u[l][i][j]=u0[l][j];
v[l][i][j]=v0[l][j];
acc[l][i][j]=acc0[l][j];
A[p][i][j]=A[p][i][j]+m1[i][l]*u[p][l][j];
B[p][i][j]=B[p][i][j]+m2[i][l]*v[p][l][j];
C[p][i][j]=C[p][i][j]+m3[i][l]*acc[p][l][j];
R1[q][i][j]=R[i][j]+A[p][i][j]+B[p][i][j]+C[p][i][j];
}}
for(int i=0;i<2;i++)
{
for(int j=0;j<1;j++)
{
u[q][i][j]=0.0;
for(int l=0;l<2;l++)
{
u[q][i][j]=u[q][i][j]+b1[i][l]*R1[q][l][j];
}
acc[p+1][i][j]=a4*(u[q][i][j]-u[p][i][j])
+a5*v[p][i][j]+a6*acc[p][i][j];
v[p+1][i][j]=v[p][i][j]+a7*(acc[p+1][i][j]
+acc[p][i][j]);
u[p+1][i][j]=u[p][i][j]+t*v[p][i][j]
+a8*(acc[p+1][i][j]+2*acc[p][i][j]);
}}}}
for(int p=0;p<=100;p++)
{
for(int i=0;i<2;i++)
{
for(int j=0;j<1;j++)
{
fout<<setiosflags(ios::fixed)
<<setiosflags(ios::showpoint)
<<setprecision(4)<<setw(20)<<u[p+1][i][j];
}
}
fout<<endl;
}
fout<<endl<<endl;
for(int p=0;p<=100;p++)
{
for(int i=0;i<2;i++)
{
for(int j=0;j<1;j++)
{
fout<<setiosflags(ios::fixed)
<<setiosflags(ios::showpoint)
<<setprecision(4)<<setw(20)<<v[p+1][i][j];
}
}
}
}

```

```

    }
}
    fout<<endl;
}
fout<<endl<<endl;
for(int p=0;p<=100;p++)
{
    for(int i=0;i<2;i++)
    {
        for(int j=0;j<1;j++)
        {
            fout<<setiosflags(ios::fixed)
                <<setiosflags(ios::showpoint)
                <<setprecision(4)<<setw(20)<<acc[p+1][i][j];
        }
    }
    fout<<endl;
}
}

```

*****INPUT DATA FILE*****

```

2
2
1
0
0
1
10001
-1
-1
1
2
1
0
0
2
1
1
1
10
0
0
0

```